

UNIVERSIDAD AUSTRAL DE CHILE

Facultad de Ciencias de la Ingeniería

Escuela de Electricidad y Electrónica



*Desarrollo de tutorial multimedial
para el diseño de proyectos con Code Composer*

Trabajo de Titulación para optar al
Título de Ingeniero en Electrónica.

Profesor Patrocinante:
Sr. Néstor Fierro Morineaud.

María Angélica Treimun Fierro

Valdivia Chile 2002

Sr. Néstor Fierro Morineaud

Sr. Franklin Castro Rojas

Sr. Raúl Urra Ríos

Valdivia, Julio de 2002

DEDICATORIA Y AGRADECIMIENTOS

Dedicada: *A Mis Padres.*

En primer lugar quiero darle las gracias a mi familia en especial a: mi papá: Roberto y a mi mamá: María, ya que gracias a su esfuerzo, apoyo, cariño y comprensión diaria pude estudiar y salir siempre adelante en cada problema que se me presentó. Además a mi hermana Marta, que se preocupó por mí y estuvo conmigo en los momentos difíciles, para enseñarme a ser mejor.

En segundo lugar quisiera darle las gracias a los profesores que siempre me respetaron y confiaron en mí y en mis capacidades como alumna en especial al Sr. Renato Loaiza, Sr. Franklin Castro y Sr. Néstor Fierro, quien fue el que me apoyó en este trabajo.

Quisiera agradecer a los grandes amigos que aquí conocí: "Jorgito N.", "Huguito V.", "Carlitos A.", "Beto M.", "Andy B.", "Tamy M.", y a "Pato B.", no sólo por *la paciencia*, que me tuvieron alguna vez en explicarme las tantas cosas que durante mi transcurso por la Universidad me costaron entender, sino también por el cariño y respeto que me han brindado siempre y por ser más que mis compañeros: Mis Amigos.

En forma especial también quiero darle las gracias a Ximena Hidalgo por ser más que mi secretaria de escuela, una persona que me brindó su confianza y su apoyo.

Por último quisiera agradecer a quienes hicieron posible el desarrollo de esta tesis como lo fue un gran colaborador Alejandro Sotomayor y Angélica Soto.

"Grachi a todos"

María Angélica Treimun Fierro.

INDICE

<u>Contenido</u>	<u>Página</u>
RESUMEN	6
SUMMARY	7
INTRODUCCIÓN	8
1. CAPITULO I: INTRODUCCIÓN A CODE COMPOSER.....	10
1.0 INTRODUCCIÓN	10
1.1 TARJETA PARA PROCESADO DE SEÑALES EN TIEMPO REAL TAC-31.	10
1.2 CODE COMPOSER STUDIO.....	13
2. CAPITULO II: MANUAL DE CODE COMPOSER.....	9
2.0 INTRODUCCIÓN	15
2.1 FILE	15
2.2 EDIT	31
2.3 VIEW	45
2.4 PROJECT	61
2.5 DEBUG	71
2.6 PROFILER	76
2.7 OPTION	83
2.8 GEL	92
2.9 TOOLS	93
2.10 WINDOW	94
2.11 HELP	94
3. CAPITULO III: DISEÑO DE PROYECTOS CON CODE COMPOSER.....	95
3.0 INTRODUCCIÓN	95
3.1 CREACIÓN DE UNA APLICACIÓN.....	96
3.2 CONFIGURACIÓN DE LA TARJETA DE TRABAJO	100
3.3 PROGRAMACIÓN EN UNA APLICACIÓN	103
3.4 GUARDANDO EL PROTECTO	105
3.5 AGREGANDO EL PROYECTO	107
3.6 VERIFICACIÓN DE LIBRERÍAS EN C.....	108
3.7 COMPILACIÓN DEL PROGRAMA	110
3.8 INCLUSIÓN DEL ARCHIVO DE COMANDOS	111
3.9 CONSTRUCCIÓN DEL PROYECTO	112
3.10 CARGANDO EL EJECUTABLE.....	115
3.11 EJECUTANDO EL PROGRAMA.....	116
3.12 CARGA DEL PROGRAMA EJECUTABLE GENERADO EN LA TARJETA TMS320c31	117

4.	CAPITULO IV: DESARROLLO DE CD MULTIMEDIAL	120
4.0	INTRODUCCIÓN	120
4.1	PROGRAMAS UTILIZADOS	120
4.1.1	DREAMWEAVER	120
4.1.2	PHOTOSHOP	122
4.2	CONTENIDO DEL CD MULTIMEDIAL	123
5.	CAPITULO V: CONCLUSIONES	128
	ANEXO A	131

RESUMEN

La nueva tecnología creciente en DSP, invade cada vez más el mercado de la electrónica; con lo cual, podemos realizar proyectos y crear utilidades prácticas que aventajen en tiempo y rapidez a las comúnmente desarrolladas; así como también adelantarse a nuevos elementos no creados aún. Es por ello que el siguiente trabajo de titulación está basado en el estudio y análisis de un software de trabajo específico con DSP. Este es el caso de "Code Composer".

Code Composer, es un software desarrollado para trabajar con la tarjeta integrada TAC 31 la que está constituida con el "DSP TMS320c31", y explicar su manejo es uno de los temas desarrollados en esta tesis, para ello se investigó a profundidad el uso de cada uno de sus comandos, explicando con ello la utilización del software sobre la tarjeta y en el desarrollo de un programa. Además gracias a la existencia de la tarjeta que trabaja con este software, TAC 31, en el Laboratorio de Procesado de Señales del Instituto de Electricidad y Electrónica de la Universidad Austral de Chile, se pudo realizar experiencias prácticas, logrando así interactuar con ella y obtener avances sobre su manejo y características de trabajo.

Una vez concluido este estudio y con la información necesaria, se procedió a crear un CD multimedial, el cual contiene un manual interactivo sobre el uso de Code Composer, preguntas frecuentes y una dirección de correo donde poder realizar consultas sobre trabajo, ya que además presenta la posibilidad de llevar su contenido a la red.

SUMMARY

The new DSP growing technology invades more and more the electronic market; with which, we can make projects and create practical usages that surpass in time and speed to the commonly developed one; in the same way to go forward to the new elements that have not been created yet. This is the reason why this thesis is based on the study and analysis of a specific software with DSP. This is the case of "Code Composer".

Code Composer is a software developed to work with the TAC 31 integrated card, which is formed with the "DSP TMS320c31", and to explain its handling is one of the topics developed in this thesis, for this the usage of each one of the commands was deeply investigated, explaining with it the software usage over the card and the development of the programme. Furthermore, due to the existence of the card that works in this software, TAC 31, at the laboratory of Electricity and Electronic Institute of the Universidad Austral de Chile, practical experiences were made, thus achieving to interact with it and obtain advances of its handling and working characteristics.

Once this study is concluded and having the necessary information, a multimedial CD was created which contains an interactive manual on the usage of the Code Composer, frequent questions and an e-mail address where questions about the job can be made, because of the fact that it also shows the possibility to take its content to the web.

INTRODUCCION

Con el rápido avance que la electrónica ha desarrollado en nuestros tiempos, se ha dado paso a elementos cada vez más poderosos y rápidos en lo que ha procesamiento de información se refiere, es así como desde que se inicio la era digital, se ha progresado en construir desde pequeños chips con poco procesamiento de información, hasta chips que aunque no aumentan mucho en tamaño, son capaces de procesar grandes cantidades de datos en pocos segundos. Esto es lo que a su paso mejora el tiempo de respuesta de cada función a realizar, entregando de esta forma respuestas de trabajo en tiempo real.

Uno de estos diseños creados para mayor rapidez en el procesamiento digital de señales son los llamados: “DSP”, que por sus siglas en inglés quiere decir: “Procesador Digital de Señales” (Digital Signal Processor).

Ahora bien, para poder crear aplicaciones y desarrollar trabajos sobre estas tarjetas se requiere básicamente de un software, instalado en el PC, que trabaje específicamente con esta tarjeta y que contenga los elementos necesarios para crear programas y visualizarlos a través de la tarjeta. Uno de estos software es: “**CODE COMPOSER**”.

Code Composer es un software creado por la Texas Instruments, TI, especializado para la creación, compilación, linkado y creación de archivos ejecutables de programas que interactúen con la tarjetas TMS320C3x y TMS320C4x. La ventaja de éste, es que es el primer software que trae incorporado los cuatro elementos (creación, compilación, linkado y creación de archivos ejecutables), en un solo programa; ya que antes, se debía realizar cada uno de estos cuatro pasos con cuatro softwares distintos, lo cual dificultaba y hacía más tedioso y lento el trabajo del usuario para interactuar con la tarjeta.

Es por ello que en el desarrollo de este trabajo de tesis, se estudió, analizó y comprendió el trabajo que realiza Code Composer, para con ello entregar una completa y detallada explicación

funcionamiento de éste, llegando finalmente a la creación de un Manual de Code Composer, el que consistió en el desarrollo de un CD multimedial, creado a través del programa "DreamWeaver 4.0" y "PhotoShop 5.0", que contiene cada uno de los pasos a seguir para la correcta utilización de Code Composer.

CAPITULO I

"INTRODUCCION A CODE COMPOSER"

1.0 Introducción.

Como ya se mencionaba con anterioridad, con los avances de la electrónica actual se han llegado a crear sistemas cada vez más sofisticados que ayudan a optimizar y minorizar el tiempo de trabajo de ciertas aplicaciones. Es por esto que en la tecnología de hoy ya se menciona constantemente el nombre de: "DSP"; pero ¿Qué es en realidad un DSP?. Un DSP es un procesador cuyo hardware y set de instrucciones están optimizados para aplicaciones numéricas de alta velocidad, algo esencial para la manipulación de datos que representan señales analógicas en tiempo real.

Pero a su vez, un sistema de Procesamiento Digital de Señales DSP, no funciona solamente con un DSP, sino además con un software que sea capaz de programar y trabajar con él. Es por ello, que en el desarrollo de este trabajo éste último será su razón fundamental.

Así como primer paso se hará una breve introducción de la tarjeta TAC-31 que es la que se encuentra disponible en el laboratorio de Procesado de Señales del Instituto de Electricidad y Electrónica de la U.A.Ch, dando las especificaciones más relevantes de ella, para después continuar con una introducción a lo que será el tema fundamental de esta tesis, el software de trabajo: "Code Composer".

1.1 Tarjeta para procesado de señales en tiempo real TAC-31.

1.1.1 Descripción General.

La tarjeta TAC31, es un tarjeta desarrollada y diseñada por la "Texas Instruments" para el Procesamiento Digital de Señales, sobre PC compatibles con bus ISA. Para ello tiene incorporado

el DSP de 32 bits en punto flotante TMS320C31, el cual es capaz de proporcionar una potencia de cálculo de 40/50 Mflops y 20/25 Mips, a una velocidad de 40 ó 50 MHz.

Por otra parte, la tarjeta dispone de dos bancos de memoria SRAM de 32/128 Kwords y 0 estados de espera, por lo que las configuraciones de memoria son 32/64/128/160/256 Kwords. Además, se dispone de una memoria de doble puerto de 2 Kwords y 1 estado de espera para las comunicaciones con el PC.

TAC-31, puede emplearse como herramienta para el desarrollo de aplicaciones de procesado de señal en tiempo real, pero además, debido a sus reducidas dimensiones y bajo costo, puede usarse como tarjeta de producción de sistemas con procesado de señales basados en PC. A tal efecto, la tarjeta dispone de una memoria no volátil en la que se pueden grabar códigos de protección de los softwares creados por el propietario.

1.1.2 Aplicaciones.

Como se dijo anteriormente, la tarjeta TAC-31 está basada en el procesador TMS320C31, que proporciona hasta 50 MFLOPS, y se encuentra equipada con hasta 1 Mbyte de SRAM sin estados de espera. Por tanto, un amplio abanico de aplicaciones de procesado de señales se puede ejecutar sobre ella, incluyendo, entre otras:

- Codificación de voz y audio.
- Reconocimiento de voz.
- Instrumentación avanzada.
- Filtrado digital.
- Síntesis de voz.
- Control adaptativo.
- Imágenes y
- Telecomunicaciones.

1.1.3 Especificaciones Técnicas.

Finalmente se presenta a continuación una tabla con las principales características técnicas correspondiente a la tarjeta TAC-31.

PROCESADO DE SEÑALES	
Procesador	TMS320C31 a 40/50 MHz
Potencia de cálculo	40/50 MFLOPS
Memoria RAM	Opcional entre 32 Kwords y 256 Kwords de 32 bits
Memoria de doble puerto	2 Kwords de 32 bits
CONEXIÓN PC	
Velocidad bus ISA	12.5 MHz
Memoria compartida	8 Kbytes
Dirección base	Seleccionable
Interrupción	Seleccionable: IRQ4/5/9/10/11/12
ENTRADA/SALIDA ANALÓGICA	
Número de canales	2
Frecuencia de muestreo	Hasta 50 KHz
Resolución	16 bits
Conectores de audio	3 Minijacks: 2 de línea y 1 para headset
FÍSICAS	
Anchura	15 mm
Altura	110 mm
Longitud	174 mm
Potencia eléctrica	10 W

Tabla 1.- Especificaciones técnicas de TAC-31.

1.2 Code Composer Studio.

Code Composer Studio es un software en un ambiente integrado de desarrollo, construido específicamente para Procesadores Digitales de Señales; DSP. Es así como en él, un usuario puede construir, corregir, perfilar y manejar proyectos en una sola aplicación dentro del mismo programa, sin necesidad de realizar varios pasos repartidos en distintos softwares, y con la ventaja de poder ser modificados, si así se desea. Además, permite realizar análisis gráficos de la señal, Entrada/Salida de los archivos, para lo cual, utiliza las herramientas de Texas Instruments en cada procesamiento de los proyectos que en él se elaboren, es decir, en definitiva es el **administrador de estas herramientas**: compilación, linkado, ensamblado y creación del ejecutable.

Code Composer es uno de los ambientes de desarrollo más sofisticados y amigables para el trabajo con DSP. Sus estructuras, presentan características únicas que incluyen capacidades adicionales para reducir el tiempo de desarrollo de cualquier aplicación.

Así, el objetivo principal de Code Composer es proporcionar las herramientas de desarrollo más avanzadas y más fáciles de utilizar para el manejo del software de DSP, aumentando con esto la productividad y reduciendo al mínimo el tiempo de desarrollo.

En la realización del siguiente capítulo se llevarán a cabo varios procesos explicativos, los cuales tienen como misión dar a conocer el manejo y utilización de Code Composer. Para ello se han desglosado en dos puntos los que pueden resumirse de la siguiente forma:

- Utilización del sistema integrado, Code Composer.
- Conocimiento en la creación de proyectos con el manejo de Code Composer.

Los Software que se emplearán provienen de tres diferentes fuentes, las que corresponden a:

Texas Instruments TI.

- Compilador C.

- Ensamblador, Montador y gestor de librerías.
- Simulador y emulador (no utilizado).

Teima Audiotex

- Librerías de rutinas de procesado de señales.
- Emulador Multiproceso.
- DspCom, paquete de comunicaciones entre procesadores.
- DspRtk, núcleo de sistema operativo de tiempo real.
- Dsplo, implementación de librería stdio de lenguaje C para el TMS320C31.
- Cargador de aplicaciones.

Y finalmente:

GO - DSP

- Code Composer V 3.04

Ya con los conocimientos necesarios sobre lo que significa un DSP y que es Code Composer, en el siguiente capítulo se procederá a explicar en detalle cada uno de los comandos que lo componen.

CAPITULO II

"MANUAL DE CODE COMPOSER"

2.0 Introducción.

El siguiente capítulo contiene el desarrollo de una guía para el usuario sobre Code Composer, el cual consta de la explicación de cada uno de los comandos contenidos en el menú principal y sus derivados. Además se detalla la funcionalidad correspondiente a cada botón de las distintas barras de herramientas, existentes en Code Composer.

Para llevar un orden en la guía de comandos se optó por comenzar de izquierda a derecha a través de la barra del menú principal, empezando por el comando "File" y su correspondiente menú, para luego seguir con el comando Edit, su menú, y así sucesivamente.

2.1 "File"

En la siguiente sección se detallarán las funciones de cada uno de los comandos de FILE, la primera función del menú principal.

2.1.1 File → New .

2.1.1.1 File → New → Source File.

Esta opción le permite al usuario crear un nuevo archivo fuente, el que no afectará la existencia de otros archivos fuentes de Code Composer.

A continuación se explican los pasos a seguir para la creación de un nuevo archivo.

- 1) Primero seleccione **File** ® **New**, del menú principal, o seleccione el botón "New" de la barra de herramientas, lo que abrirá una nueva ventana "Edit".

Botón "New": 

- 2) Luego escriba en la ventana editora su nuevo código fuente, notando que en la barra de título de la ventana Edit aparece un asterisco " * " cerca del nombre del archivo; esto indica que el archivo fuente ha sido modificado y desaparece una vez que el archivo es guardado.
- 3) Posterior a esto seleccione **File** ® **Save** o **File** ® **Save As**, del menú principal o bien presione el botón "Save File" de la barra de herramientas, con lo que abrirá la ventana de diálogo "Guardar Como".

Botón "Save File": 

- 4) En la ventana de diálogo "Guardar Como", haga doble click en el directorio donde desea guardar el nuevo archivo fuente. En caso que el directorio en que desea guardar el archivo no se encuentre visible, navegue a través de él para encontrarlo.
- 5) Así, el nombre del archivo aparecerá en el campo: "Nombre de Archivo". En caso que el usuario desee cambiar la extensión de éste, basta con que seleccione una en la lista del campo "Guardar como archivos de tipo" o de lo contrario tipear otra extensión.
- 6) Por último, cliquee "Guardar".

Los nuevos archivos son denominados como "Untitled" hasta que se les da un nombre en el momento de ser guardados.

Nota: Antes de guardar o cerrar una ventana, ésta debe estar activa. Para ello, cliquee en cualquier lugar dentro de la ventana, o seleccione **Window ® Untitled** del menú principal.

2.1.1.2 File → New → DSP/BIOS Configuration.

Este comando no es soportado en la presente edición de Code Composer.

2.1.1.3 File → New → Visual Linker Recipe.

Este comando no es soportado en la presente edición de Code Composer.

2.1.1.4 File → New → ActiveX Document.

Los documentos ActiveX (ver Anexo A) pueden ser abiertos dentro de Code Composer usando el comando **File ® New ® ActiveX Document**, del menú principal. Con esto se activará una aplicación de ActiveX, tal como Microsoft Word o Microsoft Excel.

2.1.1.4.a Como abrir un documento ActiveX dentro de Code Composer.

- 1) Del menú principal, seleccione el comando **New ® ActiveX Document**, lo que abrirá la ventana de diálogo "New Document".
- 2) Luego, seleccione una de las aplicaciones de ActiveX de la lista que aparece en la ventana de diálogo "New Document".
- 3) Finalmente cliquee OK.

Así una ventana, de una aplicación específica de ActiveX, aparecerá dentro de la ventana principal, la que puede usar para crear un nuevo documento o para abrir un documento ya existente.

2.1.2 File → Open.

Este comando sirve para abrir un archivo fuente ya existente. Para abrir este archivo, realice las siguientes acciones:

- 1) Del menú principal seleccione **File** ® **Open** o bien utilice el botón de la barra de herramientas: "File Open". Aparecerá la ventana de diálogo "Abrir".

Botón "File Open": 

- 2) En la ventana principal de "Abrir", haga doble click sobre el archivo que desea abrir. Si el archivo que desea no se encuentra visible, navegue por el directorio hasta encontrarlo y luego haga doble click sobre él.
- 3) Así, el nombre del archivo aparecerá en el campo "Nombre de Archivo". En caso que desee cambiar la extensión del archivo, basta con seleccionar una nueva extensión de la lista del campo "Guardar como archivos de tipo " o bien, tipear otra extensión.
- 4) Por último cliquee "Abrir".

2.1.3 File → Close.

Este comando cierra un archivo activo, sin salir de la aplicación. Si éste archivo contiene cambios que no han sido guardados, el usuario será advertido para guardar los cambios antes de salir.

2.1.4 File → Save.

El comando "Save", guarda un archivo con el nombre que aparece en la barra de título.

Para ello siga los pasos siguientes:

- 1) Primero que nada active el archivo cliqueando en la ventana Edit. Seleccione **File** ® **Save** o bien, use el botón "Save File" de la barra de herramientas.

Botón "Save File": 

- 2) En el caso que su archivo se encuentre sin nombre se abrirá la ventana de diálogo "Guardar Como". En el campo "Nombre de Archivo" deberá escribir el nombre que desee asignarle al archivo.
- 3) Luego, navegue al directorio en que desea guardarlo.
- 4) En el caso en que se desee cambiar la extensión del archivo, basta con tipear otra extensión o bien seleccionar uno de la lista del campo "Guardar como archivos de tipo".
- 5) Finalmente cliquee "Guardar".

2.1.5 File → Save As.

En esta parte del menú, Code Composer le ofrece al usuario la posibilidad de guardar un archivo ya existente una vez más, pero con distinto nombre, o bien cambiarle el nombre a un mismo archivo. Para esto:

- 1) Antes, active el archivo cliqueando en la ventana Edit y después seleccione **File** ® **Save As**.
- 2) Se abrirá la ventana de diálogo de "Guardar Como", y en el campo "Nombre de Archivo" deberá escribir el nombre que desee asignarle al archivo.
- 3) Luego, navegue al directorio en que desea guardarlo.

- 4) En caso que se requiera cambiar la extensión del archivo, basta con seleccionar una nueva extensión de la lista del campo "Guardar como archivos de tipo" o más bien tipear otra extensión.
- 5) Cliquee "Guardar".

2.1.6 File → Save All.

Seleccione **File** ® **Save All**, del menú principal. Esta opción guarda todos los archivos de una sola vez.

2.1.7 File → Load Program.

Para cargar un archivo COFF, (ver Anexo A), en la tarjeta del DSP, diríjase al menú principal y seleccione **File** ® **Load Program**, lo que abrirá la ventana de diálogo "Load Program". De ella seleccione el archivo deseado y cliquee "Abrir".

Este comando carga los datos así como también la información simbólica de los archivos COFF.

2.1.8 File → Load Symbol.

Para cargar una información simbólica, seleccione **File** ® **Load Symbol**. La ventana de diálogo "Load Symbol Info" se abrirá. Seleccione de ella el archivo que desea y cliquee "Abrir".

Esta funcionalidad es útil cuando el depurador no puede o no necesita cargar el archivo objeto, por ejemplo, cuando el código está en ROM. Además este comando antes de cargar uno nuevo, limpia la tabla de símbolos existentes pero no modifica la memoria.

2.1.9 File → Reload Program (Reloading a COFF File).

Para recargar un archivo COFF, en la tarjeta del DSP, seleccione desde el menú principal **File ® Reload Program**. Antes de recargar el programa, Code Composer realiza una revisión para verificar si el archivo ha sido modificado o no, desde la última carga. En caso de que no se hayan detectado cambios, sólo el programa es el que se recarga; no así su tabla de asociación de símbolos. (Además este comando es usado para recargar un programa en caso que la memoria de la tarjeta haya sido dañada).

2.1.10 File → Load Gel (Loading/Unloading GEL Functions).

Cuando se han definido archivos que contienen funciones Gel (ver Anexo A), antes de tener acceso a ellas, éstas deben ser cargadas en el archivo. Por su parte, las funciones Gel se encuentran en la memoria de Code Composer y pueden ser ejecutadas en cualquier momento, permaneciendo en la memoria hasta que sean removidas del archivo correspondiente.

Ahora bien, para que los cambios se hagan efectivos en un archivo que ha sido modificado, es necesario descargarlo y volverlo a cargar. Al momento de cargar un archivo Gel, éste, es sometido a una revisión, la cual se basa en buscar errores de sintaxis que pueda contener el archivo, no así para ver si las variables han sido definidas o no; por lo que el usuario podrá cargar una función Gel, aún antes de cargar su archivo Coff, que es el que contiene la información simbólica. Esto además corre para las funciones Gel que aún no hayan sido definidas o cargadas. Por lo tanto, los símbolos deben ser definidos cuando la función Gel sea ejecutada, así, si Code Composer encuentra un error de sintaxis mientras carga un archivo, detiene la carga y despliega el mensaje de error apropiado para el caso, y queda como misión del usuario arreglar el error antes de intentar recargar el archivo nuevamente.

2.1.10.a Como cargar un archivo Gel.

- 1) Seleccione **File ® Load Gel**, del menú principal, lo que abrirá la ventana de diálogo de "Load Gel File".

- 2) Navegue hacia el archivo que contiene las funciones Gel.
- 3) Luego en la ventana principal de la caja de diálogo, haga doble click sobre cualquier nombre de archivo o si gusta, cliquee sobre el nombre de un archivo y después presione "Abrir".

O bien:

- 1) Seleccione **View** ® **Project**, del menú principal.
- 2) Luego, cliquee con el botón derecho del mouse sobre la carpeta de "Gel files" en la ventana "Project View".
- 3) Y finalmente escoja "Load GEL" del menú contextual, para cargar un archivo Gel.

2.1.10.b Como descargar un archivo Gel.

- 1) Seleccione **View** ® **Project** del menú principal para abrir la ventana de diálogo "Project View".
- 2) Luego haga doble click sobre la carpeta para ver el contenido de archivos Gel.
- 3) Después cliquee con el botón derecho del mouse sobre los archivos de Gel que desea remover.
- 4) Finalmente seleccione "Remove" del menú contextual.

2.1.11 File → Print.

Imprime un documento. Este comando le presenta la ventana de diálogo "Imprimir", en la cual se pueden especificar las principales opciones de impresión, antes de imprimir un documento.

Botón "Print": 

Opción de teclado: Ctrl + P

2.1.12 File → Print Preview.

Antes de imprimir un documento éste debe ser presentado en la pantalla tal cual como se desee imprimir. Una vez escogido este comando la ventana principal será reemplazada por una vista preliminar del documento, en la que se exhibirán una o dos páginas en su formato de impresión. En la barra de herramientas de la vista preliminar se ofrecen las diversas opciones que se pueden escoger antes de iniciar la impresión.

2.1.13 File → Data.

2.1.13.1 File → Data → Load.

Esta opción le ofrece al usuario cargar un archivo de datos en la tarjeta del DSP a través de una dirección. Este archivo puede ser tanto un archivo objeto COFF, como un archivo de datos de Code Composer.

2.1.13.1.a Para cargar un archivo de datos.

- 1) Seleccione **File** ® **Data** ® **Load**, del menú principal, lo que abrirá la ventana de diálogo "Load Data".

- 2) Si el archivo de datos no se encuentra visible en esta ventana, navegue a través del directorio hasta encontrarlo, luego escoja el nombre del archivo y cliquee "Abrir". Esto abrirá la ventana de diálogo "Loading File into Memory".
- 3) Dentro de esta ventana, en el campo "Address", especifique la dirección donde desea que los datos sean cargados, mientras que en el campo "Length", ingrese la longitud de los datos.
- 4) Finalmente Cliquee OK.

Nota: Ambas entradas de campo son expresiones de lenguaje C.

2.1.13.2 File → Data → Save.

Code Composer le ofrece al usuario almacenar los valores de memoria de la tarjeta en un archivo de datos, el que puede ser un archivo objeto COFF o un archivo de datos de Code Composer.

2.1.13.2.a Como almacenar datos en un archivo.

- 1) Seleccione **File** ® **Data** ® **Save**, del menú principal lo que abrirá la ventana de diálogo "Store Data".
- 2) Después, especifique el nombre del archivo de datos y cliquee "Guardar", con lo que abrirá la ventana de diálogo "Storing Memory into File".
- 3) En ella, ingrese la dirección de inicio y la longitud del dato que desea almacenar.
- 4) Para finalizar cliquee OK.

Nota: Ambas entradas de campo son expresiones de lenguaje C.

2.1.14 File → Workspace.

Code Composer, le ofrece al usuario la posibilidad de restaurar y guardar su actual ambiente de trabajo. Esto lo realiza llamando al comando "Workspace", entre cada sesión de depurado. Así tiene la posibilidad de intercambiar entre distintos ambientes, en una misma sesión de depuración.

2.1.14.1 File → Workspace → Load Workspace.

Esta opción le permite al usuario cargar un workspace. Una vez en ella, siga los pasos que a continuación se presentan:

- 1) Seleccione **File** ® **Workspace** ® **Load Workspace**, del menú principal con lo que abrirá la ventana de diálogo "Load Workspace".
- 2) Luego ingrese el nombre del archivo de workspace en el campo: "Nombre de Archivo".
- 3) Finalmente cliquee "Abrir".

El usuario también tiene la posibilidad de cargar un workspace en particular cada vez que inicie Code Composer.

A continuación se presenta una lista de las cosas que pueden ser guardadas dentro de un Workspace.

- Ventana Principal (incluyendo el tamaño y la posición).
- Ventana Pequeña (incluyendo el tamaño y la posición).
- Breakpoints, Probe Points, Profile points.
- Opciones de Profiler.
- Proyecto actual.

- Funciones de Gel cargadas actualmente.
- Mapa de Memoria.
- Opciones de rapidez de Animación.
- Configuración de los archivos I/O.

Y a continuación se presenta una lista de las cosas que NO son guardadas en Workspace.

- El tipo de fuente que esté usando en esos momentos.
- El esquema de colores que esté en uso en esos momentos.
- La memoria de la tarjeta, el programa o el estado del procesador.
- Las herramientas flotantes de Edit y find/replace.
- Los mensajes de error y progreso de la construcción de un programa.
- La ventana de salida de GEL.
- La ventana de escaneo de dependencias.
- Opciones de Disassembly.

Nota: La fuente y el esquema de colores, junto con las opciones de profiler, memory map, y animate speed, son automáticamente guardadas y restauradas entre sesiones, en un archivo de nombre: "cc_user.dat".

Nota: Para iniciar el estado del procesador de la tarjeta utilice el lenguaje de extensión Gel.

2.1.14.2 File → Workspace → Save Workspace.

Este comando le permite guardar un workspace ya definido. Para ello:

- 1) Seleccione **File** ® **Workspace** ® **Save Workspace**, del menú principal. Se abrirá la ventana de diálogo "Save Workspace".
- 2) Ingrélese un nombre al "Workspace" en el campo "Nombre de Archivo".

3) Finalmente Cliquee "Guardar".

Así una vez que salga de Code Composer, su actual workspace es automáticamente guardado en un archivo con nombre "default.wks".

2.1.15 File → File I/O.

El depurador de Code Composer le permite transmitir datos que contenga un archivo del PC, desde o hacia la tarjeta del DSP. La característica de "File I/O", es que usa el concepto de "Probe Point", lo cual le permite extraer/inyectar muestras, como así tomar imágenes de locaciones de memoria en un punto definido por el mismo usuario.

Un "Probe Point" puede ser puesto en cualquier lugar del algoritmo. Cuando la ejecución de un programa alcanza a uno de ellos, el objeto conectado (ya sea un archivo, un gráfico, o una ventana de memoria), queda actualizado, reanudándose la ejecución. Usando este concepto, puede usar el comando "File I/O" si pone un "Probe Point" en un lugar específico de su código y lo conecta a un archivo.

De esta manera puede asociar el archivo con cualquier señal ya sea de entrada o salida. Así con un "Probe Point" específico y una corriente de datos, usted puede tener cualquier lectura o escritura de un archivo específico.

Nota: File I/O no soporta la transferencia de datos en tiempo real. Para mayor información sobre el uso de Transferencia de Datos en Tiempo Real (Real-Time Data Exchange (RTDX)) seleccione Help → Tools → RTDX, del menú principal.

2.1.15.a Para transferir datos desde o hacia un archivo.

1) Antes que nada el usuario deberá posicionar el cursor en el punto sobre el cual desea colocar un "Probe Point". Luego presione el botón "Toggle Probe Point" de la barra de herramientas de Project, y deje el "Probe Point" desconectado. De esta forma, el "Probe

Point" le informará al depurador cuando desea que se inicie la transferencia de datos, desde o hacia el archivo. En otras palabras una vez que la ejecución del código alcanza este punto, el depurador actualiza (o lee desde) el archivo que está conectado al "Probe Point". Una vez terminado esto, la ejecución continua.

Botón "Toggle Probe Point": 

- 2) Luego, seleccione **File** ® **File I/O** del menú principal. Se abrirá la ventana de diálogo "File I/O". Escoja entre las etiquetas "File Input" o "File Output", según sea el caso.
- 3) Posteriormente, presione el botón "Add File", ya sea en la etiqueta de File Input o en la de File Output, lo que abrirá la ventana de diálogo de File Input (o de File Output, según sea el caso escogido).
 - Una vez abierta la ventana, basta con navegar en el directorio en busca de la carpeta que contiene el archivo que desea.
 - Aparecerá, entonces, en la ventana de diálogo, el nombre de los archivos que contienen las carpetas. De ellos seleccione uno, apareciendo su nombre en el campo "Nombre de Archivo". Los datos éste, pueden ser del tipo COFF o un archivo de datos de Code Composer.
 - Posterior a ello cliquee "Abrir". Ahora el nombre del archivo aparecerá en la ventana de diálogo "File I/O". En caso de que desee añadir más archivos basta con repetir el procedimiento anterior, cliqueando "Add File", en cualquiera de las dos etiquetas.

Después al agregar un archivo, aparece una ventana "File I/O control", la que le permitirá monitorear y controlar en forma activa el progreso del archivo.

- 4) Además, cuando se agrega un archivo a través de la ventana de diálogo "File I/O", éste no se conecta directamente a un "Probe Point", ya que si se observa bien el campo de: "Probe Point" muestra la palabra: "Not Connected"; por lo tanto para conectar un

archivo a un "Probe Point", cliquee en el botón "Add Probe Point". La ventana de diálogo "The Break/Probe/Profile Points" aparecerá, trayendo con ella la etiqueta "Probe Points", ya seleccionada.

- En la lista "Probe Point", se presenta con anticipación el "Probe Point" que se desea conectar. (Sin embargo, este punto aún se encuentra con la opción "No Connection").
- Si el "Probe Point" que se presenta no es el que se requiere, entonces escoja el adecuado en la lista "Connect To".
- Luego cliquee en: "Replace" y vea que la lista de "Probe Point" muestra que éste ya está conectado al archivo seleccionado.
- Cliquee OK y observe como el campo de: "Probe", en la ventana de diálogo "File I/O", muestra la palabra "Connected" cada vez que un archivo se ha conectado a un "Probe Point" .

5) Por su parte en la ventana de diálogo "File I/O", han sido ingresados los valores de los campos: "Address" y "Length" (para cada uno de los archivos seleccionados).

El campo: "Address" especifica, donde desea que los datos sean transferidos, ya sea hacia (File Input) o desde. (File Output). En este campo, se pueden ingresar cualquier nombre o dirección numérica válidas.

Ahora bien, el campo "Longitud" indica cuántas muestras son transferidas desde (File Output) o hacia (File Input) la tarjeta cada vez que el "Probe Point" es alcanzado.

Además usted puede ingresar en los campos "Address" y "Length" cualquier expresión de C (válida). Estas expresiones son recalculadas cada vez que las muestras son leídas desde la tarjeta o escritas en ella, lo que significa que si se ingresa un símbolo en este campo, que después cambia en valor, no tiene que reingresar este parámetro.

6) Finalmente cliquee "Aceptar". Los parámetros ingresados, serán confirmados.

2.1.15.b Mode Wrap Around.

En la etiqueta de File Input, usted puede seleccionar el modo "Wrap Around", el cual realiza un loop a un archivo, de manera que cuando llegue al final de éste, vuelva a él comenzando desde arriba. Esta característica es útil cuando se desea generar una señal periódica desde un archivo. Por otro lado si la opción de modo Wrap Around no es seleccionada y el final del archivo es alcanzado, el usuario es advertido a través de un mensaje en pantalla, el que indica que el archivo a llegado a su final y que el programa del DSP será detenido.

2.1.16 File → Recent Source File.

Para reabrir un archivo fuente guardado recientemente, sólo debe seleccionar la opción **File** ® **Recent Source Files**, del menú principal y escoger el nombre del archivo que busca.

2.1.17 File → Recent Workspace.

Para cargar un área de trabajo guardada recientemente, sólo debe seleccionar la opción **File** ® **Recent Workspaces**, del menú principal y escoger el nombre del archivo que desea .

2.1.18 File → Recent Program File.

Para volver a un archivo de programa guardado recientemente, sólo debe seleccionar la opción **File** ® **Recent Program Files**, del menú principal y escoger el nombre del archivo que desea.

2.1.19 File → Exit.

Para finalizar su sesión con Code Composer, puede seleccionar **File** ® **Exit**, del menú principal como también pulsar el botón "Close".

Botón "Close": 

O bien

Presione las teclas: ALT + F4

2.2 "Edit"

En la siguiente sección se detallarán las funciones de cada uno de los comandos de EDIT, la segunda función del menú principal.

2.2.1 Edit → Undo.

Para deshacer la última acción realizada dentro de la ventana activa seleccione **Edit** ® **Undo**, del menú principal; o si lo desea también puede usar el teclado presionando las teclas: **Ctrl** + **Z** o bien el botón "Undo" de la barra de herramientas.

Botón "Undo": 

Nota: No todas las acciones pueden deshacerse.

2.2.2 Edit → Redo.

Para rehacer la última acción dentro de la ventana activa, seleccione **Edit** ® **Redo** del menú principal, o también puede usar el teclado presionando las teclas: **Ctrl** + **Y** o el botón "Redo" de la barra de herramientas.

Botón "Redo": 

2.2.3 Edit → Cut.

Para quitar un texto seleccionado de la ventana activa, realice una de las siguientes acciones: seleccione el comando **Edit** ® **Cut**, del menú principal, o si lo desea utilice el botón "Cut" de la barra de herramientas o bien use el teclado presionando las teclas: **Mayús + Supr**, con lo que copiará el contenido en el portapapeles para su uso posterior.

Botón "Cut": 

2.2.4 Edit → Copy

Ahora bien, si desea "Copiar" un texto seleccionado, utilice el comando **Edit** ® **Copy** del menú principal, el que copiará el contenido en el portapapeles para su uso posterior. También puede utilizar el teclado presionando las teclas **Ctrl + C** o si lo desea el botón de la barra de herramientas: "Copy".

Botón "Copy": 

2.2.5 Edit → Paste.

Luego; si desea insertar el texto contenido en el portapapeles, utilice el comando **Edit** ® **Paste**, del menú principal. También puede utilizar el teclado presionando las teclas **Ctrl + V**, o si lo desea el botón de la barra de herramientas: "Paste".

Botón "Paste": 

2.2.A Como Cortar, Copiar, y Pegar un texto.

- 1) Resalte el texto que quiere cortar o copiar.

- 2) Seleccione **Edit** ® **Cut** o **Edit** ® **Copy**. También puede usar los botones de la barra de herramientas:

Botón "Cortar": 

Botón "Copiar": 

- 3) Sitúe un punto de inserción en cualquier parte de la ventana de Edición, donde quiera colocar el texto.

- 4) Finalmente seleccione **Edit** ® **Paste**, sino use el botón de la barra de herramientas:

Botón "Pegar": 

2.2.6 Edit → Delete.

Esta acción anula el texto resaltado sin copiarlo al portapapeles, por lo que el usuario no puede pegar este texto a otro lugar.

Botón: "Delete" o "Supr" del teclado.

2.2.7 Edit → Select all.

Este comando selecciona todo el texto contenido en la ventana activa. Para ello seleccione desde el menú principal **Edit** ® **Select All** o simplemente presione las teclas **Ctrl + A**, del teclado.

2.2.8 Edit → Find/Replace .

Esta opción le permite al usuario, además de encontrar un texto dentro de un archivo, buscar una serie de textos y reemplazarlo por otro.

2.2.8.a Para Encontrar y Reemplazar un texto.

- 1) Posicione el punto dónde quiere empezar su búsqueda.
- 2) Seleccione **Edit** ® **Find/Replace** del menú principal. Aparecerá la ventana de diálogo de "Find and Replace".
- 3) Típee el texto que desea buscar en el campo: "Find".
- 4) Luego típee en el campo: "Replace" el nombre del texto con el que desea reemplazar al texto anterior.
- 5) A continuación presione el botón: "Find Next".
- 6) Una vez encontrado el texto presione el botón: "Replace" para reemplazar el texto seleccionado.

Como otra alternativa, presione el botón: "Replace All" para reemplazar todos los textos del párrafo seleccionado.

Además, el usuario podrá controlar la búsqueda de la ventana de diálogo Find/Replace, haciendo click en el botón "Properties" y escogiendo las opciones que desea determinar para su búsqueda.

2.2.9 Edit → Find in File.

El comando **Edit** ® **Find in File** del menú principal, le permite buscar y encontrar alguna palabra o expresión, dentro de los archivos de texto existentes.

2.2.9.a Como encontrar una palabra o una expresión, dentro de los archivos.

1. Del menú principal, seleccione el comando **Edit** ® **Find in File**, con el cual se abrirá la ventana de diálogo "Find in Files". O bien, puede usar el botón de la barra de herramientas: "Find in File".

Botón "Find in File": 

2. Una vez en la ventana, entregue la siguiente información en cada uno de los campos:
 - **Find What.** En este campo ingrese la palabra o la expresión que desea buscar, o bien use la lista que se presenta en este campo para escoger de ella una palabra buscada con anterioridad.
 - **Files of type.** En este campo seleccione el tipo de archivo que desea buscar, ya sea, tipeando en el campo la extensión de éste o bien seleccionando una de las extensiones que se presentan su la lista.
 - **In Folder.** Aquí, seleccione la carpeta por la cual desea iniciar su búsqueda, o bien utilice el botón: "..." para navegar a través del directorio y seleccionar la carpeta que desea.
 - **Look in subfolder.** Marque esta opción, si quiere que también se busque en las carpetas que están dentro de la carpeta escogida como inicial, en el campo anterior.
 - **Match case.** Marque esta opción para buscar textos que se relacionen con la palabra o expresión ingresada para su búsqueda.
 - **Match Whole word only.** Esta opción habilita la posibilidad de buscar un texto que se relacione sólo con las palabras que NO son precedidas ni seguidas por un carácter alfanumérico ni por un guión abajo: "_".
 - **Look In Project Files.** Marque esta opción para realizar la búsqueda dentro de un proyecto en particular.
3. Cliquee Find para iniciar la búsqueda.

Una vez finalizada, se despliega una ventana de salida, "Output", que contiene los resultados obtenidos. En ella, se presenta, para cada palabra encontrada, la ruta del archivo en la que se encuentra, seguido por el número de línea y la palabra que contiene esta línea (que se supone es la palabra buscada).

Luego, si desea abrir cualquiera de los archivos encontrados como resultado de la búsqueda, basta con hacer doble click sobre él y se abrirá una ventana editora con el archivo especificado; donde el cursor se ubicará al inicio de la línea que contiene la expresión escogida.

Para cerrar la ventana de salida: "Output", cliquee con el botón derecho del mouse sobre ella y seleccione la opción: "Hide" del menú contextual.

2.2.10 Edit → Go To.

Este comando le permite ir a una línea específica o a un bookmarks dentro de un archivo fuente rápidamente.

2.2.10.a Como ir a una línea específica o a un Bookmark.

- 1) Seleccione el comando **Edit** ® **Go**, del menú principal, con lo que abrirá la ventana de diálogo "Go To".

O bien

Cliquee con el botón derecho del mouse dentro de la ventana "Edit" y seleccione: "Go To" del menú contextual.

- 2) Luego especifique la línea o bookmark que desea ver.
- 3) Finalmente cliquee OK.

2.2.11 Edit → Memory.

2.2.11.1 Edit → Memory → Edit.

Code Composer también le permite al usuario editar un lugar de la memoria. Para ello:

- En la ventana de memoria; haga doble click sobre la ubicación de memoria que desea editar, o bien
- Seleccione el comando **Edit** ® **Memory** ® **Edit**.

Ambos métodos le abrirán la ventana de diálogo: "Edit Memory".

En el primer caso, si el usuario ha realizado un doble click en la ubicación de memoria que desea editar, se abrirá una ventana con dos campos: "Address" y "Data", los que contienen los valores de la dirección y los datos de la ubicación de memoria seleccionada.

En el segundo caso, si se utilizó el comando del menú para abrir la ventana de diálogo, los campos "Address" y "Data" contendrán los valores dados por defecto; por lo tanto para editar una dirección de memoria deseada, el usuario deberá ingresar dicha dirección en el campo "Address". Luego deberá hacer click sobre el campo "Data". Éste, desplegará automáticamente su contenido, el cual se encuentra actualizado con el valor de los datos, que contenga la dirección especificada. En caso de querer cambiar el valor de estos datos, sólo tiene que ingresar el valor que desea en el campo "Data" y luego presionar "Done". Además el usuario podrá usar el scroll (o botón giratorio, de los mouses que lo contienen), para moverse a través de las ubicaciones de memoria.

Por otra parte si se realiza un doble click, sobre alguna ubicación de la ventana "Memory", por defecto, el valor que aparece en el campo "Data" siendo el mismo valor sobre el cual se hizo doble click en la ventana. Así, el usuario podrá ingresar valores **enteros** en ambos formatos, tanto decimal (sin prefijo) como hexadecimal (anteponiendo el prefijo 0x), como además valores enteros de punto flotante, el cual es provisto de un formato compatible con él.

Nota: Todos los campos de entrada son expresiones en C

2.2.11.2 Edit → Memory → Copy.

Si el usuario desea copiar un bloque de memoria a una nueva ubicación, deberá realizar los siguientes pasos:

- 1) Seleccione **Edit** ® **Memory** ® **Copy** del menú principal, con lo que abrirá la ventana de diálogo de: "Setup for Copying".
- 2) Ingrese la información siguiente.

En el campo de:

- **Address.** Ingrese la dirección de inicio del bloque de memoria a ser copiado.
- **Length.** Ingrese la longitud del bloque de memoria a ser copiado.

- 3) Ingrese la información de destino.

En el campo de:

- **Address.** Ingrese la dirección a la cual el bloque de memoria debe ser copiado.

- 4) Por último cliquee OK para realizar la copia.

Nota: Todos los campos de entrada son expresiones en C.

2.2.11.3 Edit → Memory → Fill

Esta opción sirve en el caso que el usuario desee llenar un bloque de memoria con un valor determinado, para ello deberá seguir los pasos que a continuación se presentan:

1) Primero seleccione **Edit** ® **Memory** ® **Fill** desde el menú principal, el que abrirá la ventana de diálogo "Setup Filling Memory".

2) Luego, ingrese la siguiente información:

En el campo de:

- **Address.** Ingrese la dirección de inicio, del bloque de memoria que será llenado.
- **Length.** Ingrese la longitud del bloque de memoria que será llenado.
- **Fill Pattern.** Ingrese el patrón a usar en el llenado del bloque de memoria.

3) Cliquee OK.

Todos las ubicaciones son rellenas, a partir de la dirección de inicio hasta "la dirección de inicio + la longitud - 1", de acuerdo al patrón de relleno ingresado en el campo correspondiente.

Nota: Todos los campos de entrada son expresiones en C.

2.2.11.4 Edit → Memory → Patch Asm.

La característica principal de Patch assembly es que le permite modificar rápidamente, el código ejecutable de la tarjeta del DSP sin tener que reconstruir el proyecto. Para llamar esta opción, seleccione del menú principal **Edit** ® **Memory** ® **Patch ASM** o bien, en la ventana de Dis-Assembly, posicione el cursor sobre la dirección de interés, luego presione el botón derecho del mouse y escoja la opción "Patch Assembly" del menú contextual. La ventana de diálogo "Patch Assembly" le presentará las siguientes opciones:

- **Address.** Este campo, contiene la dirección de las instrucciones que se desean modificar y es automáticamente incrementado cada vez que sus instrucciones son

ensambladas con éxito. Por su parte el scroll del mouse, en caso que lo contenga, puede ser usado en el campo de "Address", para desplazarse a través de los puntos de memoria en forma manual.

Nota: Todos los campos de entrada son expresiones en C.

- **ASM Instruction:** En este campo el usuario podrá ingresar cualquier expresión (válida) del DSP, incluyendo símbolos COFF. Una vez ingresada la instrucción, basta con clicar "Apply" para iniciar la ejecución de Patch. Si todo anda bien, y las instrucciones son ensambladas sin errores, el campo "Address" se actualiza para la próxima dirección disponible del programa.
- **Status :** Después de haber hecho click sobre "Apply", se desplegará un mensaje en el campo Status describiendo los resultados de Patch Assembly.

2.2.12 Edit → Edit Register.

A continuación se detallan los pasos para editar el contenido de un registro.

- 1) Desde el menú principal seleccione el comando **Edit ® Edit Register**. Lo que abrirá la ventana de diálogo: "Edit Registers".

O bien

- De la ventana "Register", haga doble click sobre un registro o cliquee el botón derecho del mouse y seleccione "Edit Register" del menú contextual.

Una vez en la ventana de diálogo "Edit Registers" usted podrá escoger entre las siguientes opciones:

- **Register.** Típee en este campo, el nombre del registro que desea editar o bien seleccione un registro de la lista presentada en este campo.

- **Value**. Este campo contiene el valor actual en hexadecimal del registro especificado, sin embargo el usuario también tiene la posibilidad de ingresar otro valor distinto al contenido en este campo. Para ello debe considerar que si el valor a ingresar está en hexadecimal debe agregar el prefijo "0x" y si está en decimal pues no lleva prefijo alguno.

Nota: Todos los campos de entrada son expresiones en C.

Después de modificar los valores de un registro, cliquee OK para guardar los cambios y para cerrar la ventana de diálogo cliquee Close.

2.2.13 Edit → Edit Variable.

Este comando le permite editar una variable, para ello siga los pasos que a continuación se detallan.

- 1) Select **Edit** ® **Edit Variable** del menú principal. Seguido a ello aparecerá la ventana de diálogo "Edit Variable".
- 2) Luego ingrese la siguiente información:

En el campo:

- **Variable**: Ingrese el nombre de la variable a editar.
- **Value**: Ingrese el nuevo valor.

- 3) Finalmente cliquee OK.

La ventana de diálogo "Edit Variable" también puede ser usada para la edición de expresiones cuando se trabaja en la ventana "Watch".

Nota: Todos los campos de entrada son expresiones en C.

2.2.14 Edit → Edit Command Line.

Este comando provee una manera muy conveniente de ingresar o ejecutar expresiones de Gel. De esta manera el usuario podrá ejecutar cualquiera de estas funciones incorporadas en Code Composer, como también podrá ejecutar sus propias funciones de Gel que hayan sido cargadas.

2.2.14.a Como ejecutar comandos de Gel.

- 1) Seleccione **Edit** ® **Edit Command Line** del menú principal, el que abrirá la ventana de diálogo "Command Line".
- 2) Ingrese en el campo "Command", una expresión o función GEL.
- 3) Por último cliquee "OK" para ejecutar los comandos.

También puede acceder a estas funciones a través de la barra de herramientas de Gel, seleccionando **View** ® **GEL Toolbar**. En ella se puede ver el campo de archivos Gel y una lista de las funciones más recientes de Gel ejecutadas.

Para ejecutar uno de los comandos ya previamente usados, basta con seleccionarlo y presionar el botón de ejecución.

Botón Execute: 

A continuación se presentan algunos ejemplos de comandos que pueden ser ingresados tanto en la barra de herramientas de Gel como en el diálogo "Command Line" :

- a) Para modificar variables ingresando expresiones:

PC = c_int00

- b) Para cargar programas con las funciones de Gel que viene incorporadas:

GEL_Load("c:\myprog.out")

- c) Para hacer correr sus propias funciones de Gel:

MyFunc()

2.2.15 Edit → Column Editing.

Code Composer además ofrece la posibilidad de seleccionar, cortar y pegar columnas de texto en lugar de filas.

Como primer paso se debe activar una fila, cliqueándola en la ventana "Edit". Luego se debe ingresar al modo "Column" seleccionando **Edit** ® **Column Editing** o presionando las teclas: **Ctrl** + **Shift** + **F8**, luego se mueve el cursor a la columna que se desea seleccionar, se hace click y se arrastra para seleccionar dicha columna.

Otra forma de seleccionar una columna es presionando la tecla Shift e ir moviendo el cursor con las flechas del teclado. Así, el usuario puede cortar, copiar, pegar y borrar tantas columnas como las que desee seleccionar.

2.2.16 Edit → Bookmarks.

Esta opción le ofrece al usuario colocar bookmarks, los que sirven para encontrar y mantener ubicaciones claves dentro de sus archivos fuentes (como por ejemplo: colocar un bookmark sobre una línea que se deba corregir, con la frase: "corregir", en el campo "Description", como se verá más adelante). Éstos pueden ser puestos sobre cualquier línea en cualquier archivo, y son guardados en el área de trabajo de Code Composer, a fin de que puedan ser ubicados en cualquier momento.

2.2.16.a Cómo poner un Bookmark desde la ventana Edit.

Para poner un bookmark mientras se edita un código fuente en la ventana Edit, se deben seguir los siguientes pasos:

- 1) Posicione el cursor sobre la línea en la que desea colocar un bookmark.
- 2) Luego, cliquee con el botón derecho del mouse sobre la ventana Edit y seleccione "Bookmarks" del menú contextual. Ahora, desde el submenú de Bookmarks, seleccione "Set a Bookmark".

O bien,

Presione el botón " Edit: Toggle Bookmark" de la barra de herramientas de Edit.

Botón "Edit: Toggle Bookmark": 

(Note que la línea donde está ubicado el bookmark es destacada en la ventana Edit).

Ahora bien, para avanzar rápidamente desde un bookmarks a otro utilice los botones "Edit: Next Bookmark" y " Edit: Previous Bookmark" de la barra de herramientas de Edit.

Botón "Edit: Next Bookmark": 

Botón "Edit: Previous Bookmark": 

2.2.16.b Para ver el listado de Bookmarks .

Para ver el listado de Bookmarks, utilice cualquiera de los pasos que a continuación se presentan:

- Seleccione la etiqueta Bookmarks en la ventana Project View. Luego cliquee sobre cualquiera de los bookmarks que allí aparecen con lo que abrirá el archivo que contiene al bookmark seleccionado. Luego posicione el cursor en la ubicación de el bookmark.
- Seleccione el comando **Edit** ® **Bookmarks** para abrir la ventana de diálogo de Bookmarks.

O bien

Presione el botón "Edit: Bookmarks", de la barra de herramientas de Edit, lo que abrirá la ventana de diálogo "Bookmarks".

Botón "Edit: Bookmarks": 

2.3 "View"

En la siguiente sección se detallarán las funciones de cada uno de los comandos de "VIEW", la tercera función del menú principal.

2.3.1 View → Standard Toolbar.

Con este comando se puede seleccionar el ver o no la barra de herramientas Standard, la que permite el rápido acceso con el mouse a herramientas que se usan con frecuencia. Se presenta en la parte superior de la ventana de aplicación, debajo de la barra del menú principal.



Ya sea para ocultar o desplegar la barra de herramientas, seleccione **View** ® **Standard Toolbar** o bien **Alt + V + T**.

Cliquee: Para:



Abrir un nuevo documento.



Abrir un documento ya existente. (Se abre la ventana Open en la que el usuario busca y abre el archivo que desee).



Guardar el documento activo o el que se ha creado con el nombre que esté usando. En caso de que no tenga nombre aún, se abrirá la ventana Save As.



Imprimir el documento actual.



Remover el dato seleccionado en el documento y lo almacena en el portapapeles.



Copiar la selección al portapapeles.



Insertar o "pegar" el contenido guardado en el portapapeles, en el punto de inserción escogido.



Deshacer la última acción.



Rehacer la acción previamente deshecha.



Encontrar el **próximo** texto que coincida con el texto especificado. (Busca las: palabras, sílabas o letras siguientes a la actual, que sean iguales a las seleccionadas dentro de la ventana "Edit").



Encontrar el texto **anterior** que coincida con el texto especificado. (Busca las: palabras, sílabas o letras anteriores a la actual, que sean iguales a las seleccionadas dentro de la ventana "Edit").



Encontrar el texto especificado en el archivo especificado. (Ver: 2.9.a Como encontrar una palabra o una expresión, dentro de los archivos.)



Buscar una palabra específica o una sección de texto, dentro de la ventana "Edit".



Llamar al contexto de ayuda.

2.3.2 View → GEL Toolbar

La barra de herramientas Gel, sirve para acceder y ejecutar las funciones de GEL.



Para desplegar u ocultar esta barra, seleccione del menú principal: **View ® GEL Toolbar** o bien presione las teclas: **ALT + V + G**.

Cliquee: Para:



Ejecutar la función especificada.

2.3.3 View → Project Toolbar.

La barra de herramientas de Project proporciona un rápido acceso con el mouse a los comandos usados en la construcción de un proyecto y a los usados para colocar Breakpoints, Probe Points, y Profile Points.



Para desplegar u ocultar esta barra, seleccione del menú principal: **View** ® **Project Toolbar** o bien presione las teclas **Alt + V + O**.

Cliquee: Para:



Compilar el archivo fuente en uso.



Construir paso a paso el proyecto.



Construir el proyecto entero, ignorando las dependencias.



Detener la construcción que se realiza en esos momentos.



Poner un Breakpoint en la ubicación actual.



Remover todos los Breakpoints.



Colocar Probe Points en la ubicación actual.



Remover todos los Probe Points.



Colocar Profile Points en la ubicación actual.



Remover todos los Profile Points.

2.3.4 View → Debug Toolbar.

La barra de herramientas de Debug entrega un rápido acceso con el mouse a los comandos más usados de ella.



Para desplegar u ocultar esta barra, seleccione del menú principal: **View** ® **Debug Toolbar** o bien presione las teclas **Alt + V + B**.

Cliquee Para:



Step Into, sitúa el cursor dentro de una parte del programa y avanza a través de las instrucciones del programa de una en una.



Step Over, es usado para avanzar paso a paso a través de la función actual y para ejecutar órdenes individuales en ella.



Step Out, usado para completar la ejecución de una función fuera del programa (como por ejemplo una subrutina). Así la ejecución se detiene, después que la función actual retorna al llamado de la función.



Hacer correr un programa hasta que encuentre la posición del cursor en la ventana de Dis-Assembly.



La opción Run, la que sirve para ejecutar o "echar a correr" el programa actualmente en uso.



La opción Halt, que sirve para detener el programa actualmente corriendo.



Animar la tarjeta, es decir agregarle diversos puntos de análisis (Probe Point, Profile Point, Breakpoint) al programa mientras está corriendo, para ver y analizar su ejecución.



Ver la ventana Quick Watch



Ver la ventana Watch

-  Ver la ventana CPU Registers
-  Ver la ventana Memory
-  Ver la ventana de Call Stack
-  Ver la ventana de Disassembly

2.3.5 View → Edit Toolbar.

La barra de herramientas "Edit" entrega un rápido acceso a los comandos usados para edición y para el acceso de bookmarks.



Esta barra es desplegada automáticamente al iniciarse Code Composer.

Ahora bien en caso que no se despliegue o al contrario quiera ocultarla, seleccione **View** ® **Edit Toolbar**, del menú principal.

Cliquee Para:

-  Marcar el texto contenido entre un paréntesis o llave abierta y su correspondiente paréntesis o llave cerrada, cuando se posiciona el cursor un lugar antes de éste paréntesis abierto o llave abierta..
-  Buscar y marcar el **próximo** texto contenido entre un paréntesis abierto o llave de inicio y su correspondiente paréntesis o llave de cierre.
-  Mover el cursor entre un paréntesis o llave abierta y su correspondiente paréntesis o llave cerrada, dentro de un texto del programa.



Mover el cursor hacia el próximo paréntesis abierto o llave abierta.



Mover un bloque de texto seleccionado un paso a la izquierda hasta la detención del tabulador.



Mover un bloque de texto seleccionado un paso a la derecha hasta la detención del tabulador.



Crear o eliminar un bookmark dentro de un documento activo, dependiendo de la línea actual en la que se encuentre posicionado.



Encontrar el bookmark siguiente en el documento activo.



Encontrar el bookmark anterior en el documento activo.



Abrir la ventana de diálogo de Bookmarks.

Además se puede acceder a esta barra de herramientas cliqueando con el botón derecho del mouse y seleccionando **Tools** ® **Edit Toolbar**.

2.3.6 View → Status Bar

La barra de estado es aquella que se encuentra desplegada bajo la ventana de aplicación principal.

Para desplegar u ocultar esta barra, seleccione del menú principal: **View** ® **Status Bar**. En caso que se haya escogido la opción de desplegar esta barra, aparecerá una marca en el menú, al lado de este comando.

Al deslizar el cursor sobre los comandos o al usar las flechas del teclado para navegar a través del menú, se describe en el área izquierda de esta barra; las acciones que cada uno de estos comandos realizan. Además muestra mensajes que describen las acciones de los botones de las barras de herramientas cuando se posiciona el cursor sobre alguno de ellos.

Por su parte, el área derecha de esta barra, indica cual de las siguientes teclas, que a continuación se mencionan, se encuentran activas:

Indicador	Descripción
CAP	Caps Lock se encuentra activada.
NUM	Num Lock. se encuentra activada
SCRL	Scroll Lock se encuentra activada.

2.3.7 View → Dis-Assembly.

Cuando se carga un programa en la tarjeta del DSP o simulador, el depurador de Code Composer automáticamente abre una ventana "Dis-Assembly".

La ventana Dis-Assembly despliega las instrucciones desambladas y la información simbólica necesaria para el depurado. Disassembly revierte el proceso de ensamblaje y permite que el contenido de la memoria sea desplegado como un código de lenguaje Assembler. Por su parte, la información simbólica consiste en símbolos y series de caracteres alfanuméricos que representan direcciones o valores de la tarjeta del DSP.

Ahora bien, para cualquier instrucción en lenguaje assembler, esta ventana presenta el "desamblado" de ella, la dirección en la cual está ubicada y el opcode correspondiente. (opcode: código máquina que representa la instrucción). Para realizar el desamblado de instrucciones, el depurador lee los opcode de la tarjeta, los "desambla", y adiciona la información simbólica comenzando en la ubicación indicada por el contador de programa activo: PC. La línea que contiene el PC actual, es destacada en color amarillo.

Nota: Para las secciones del programa que están escritas en lenguaje C, el usuario puede escoger la opción **View** ® **Mixed Source/Asm** del menú principal, que le permite ver tanto el código fuente escrito en C, como su código en assembler.

2.3.8 View → Memory.

El depurador de Code Composer le permite al usuario ver, en una ubicación específica de la memoria, el contenido de ésta.

2.3.8.a Para ver el contenido de la Memoria.

- 1) Seleccione **View** ® **Memory**, del menú principal.

O bien

Seleccione el botón View Memory de la barra de herramientas de Debug.

Botón "Memory Window": 

- 2) Antes de abrirse la ventana "Memory", se abrirá la ventana "Memory Window Options", la que le permitirá al usuario escoger las características de la ventana de Memory. Una vez ahí, ingrese las características que desea.

O bien

Dentro de la ventana "Memory", cliquee con el botón derecho del mouse sobre ella y seleccione "Properties" del menú contextual, con lo cual abrirá la ventana de diálogo de "Memory Window Options".

- 3) Cliquee OK. Se abrirá la ventana "Memory".

Si requiere editar el contenido de una ubicación de la memoria, basta con hacer doble click en la dirección apropiada de la ventana "Memory" o seleccionar **Edit** ® **Memory** ® **Edit**, apareciendo con esto la ventana de diálogo "Edit Memory", en la que se podrán editar los datos que se requieran.

2.3.9 View → CPU Registers.

2.3.9.1 View → CPU Registers → CPU Registers.

Para ver el contenido de los registros (de la CPU) del procesador de la tarjeta, se debe seleccionar del menú principal el comando **View** ® **CPU Registers** ® **CPU** o bien, presionar el botón de la barra de herramientas de Debug: "Register Window". De esta forma el usuario puede editar los registros a través de la ventana de diálogo de Edit Registers.

Botón Register Window: 

2.3.10 View → Graph.

Code Composer incorpora una avanzada interfaz de análisis de señal, que permite monitorear con exhaustividad las señales de dato. Además es muy útil en el desarrollo de aplicaciones para comunicaciones, tales como: wireless, procesamiento de imagen y en aplicaciones generales de DSP.

A continuación se presentan los 4 tipos de gráficos disponibles para utilizar dentro de Code Composer.

2.3.10.1 View → Graph → Time/Frequency.

El menú de Gráfico presenta diversas opciones que el usuario puede elegir para ver sus datos. La opción "Time/Frequency", como su nombre lo indica, le permite ver la señal analizada ya sea en el dominio del tiempo o la frecuencia.

Para el análisis de una señal en el dominio de la frecuencia, el buffer que se despliega, se hace correr a través de una rutina FFT. Los gráficos en este dominio incluyen: FFT Waterfall, Complex FFT, y la Magnitud y Fase de FFT.

En el dominio del tiempo los gráficos pueden ser Simples o Dobles. Para ver el gráfico Time/Frequency seleccione **View ® Graph ® Time/Frequency**, del menú principal, lo que abrirá la ventana de diálogo "Graph Property". En ella podrá visualizar una columna izquierda que contiene el nombre de cada campo, y una derecha en la que podrá ajustar los valores de cada uno de estos campos; teniendo con ello la posibilidad de ajustar los valores que le sean necesarios. Una vez hecho esto basta con clicar "OK" y la ventana de gráfico aparecerá con los valores escogidos. En caso de que el usuario desee cambiar alguno de estos valores deberá clicar sobre la ventana de gráfico con el botón derecho del mouse, seleccionar "Properties", y ajustar los parámetros que desee. También puede actualizar el gráfico en cualquier punto de su programa.

Todos los campos de entrada son expresiones C.

2.3.10.2 View → Graph → Constellation.

Este gráfico es usado para medir como, efectivamente, la información es extraída de la señal de entrada, la cual es separada en dos componentes y el dato resultante es trazado usando el sistema de coordenadas Cartesianas en el tiempo, trazando una señal versus la otra (Fuente Y versus Fuente X , donde Y es trazada sobre el eje Y y X sobre el eje X).

Para visualizar este tipo de gráfico, seleccione desde el menú principal: **View** ® **Graph** ® **Constellation**, el que abrirá la ventana de diálogo "Graph Property". Al igual que el gráfico anterior podrá visualizar una columna izquierda que contiene el nombre de cada campo, y una derecha en la que podrá ajustar los valores de cada uno de estos campos; teniendo con ello la posibilidad de ajustar los valores que le sean necesarios. Una vez hecho esto basta con clicar "OK" y la ventana de gráfico aparecerá con los valores escogidos. En caso de que el usuario desee cambiar alguno de estos valores deberá clicar sobre la ventana de gráfico con el botón derecho del mouse, seleccionar "Properties", y ajustar los parámetros que desee. También puede actualizar el gráfico en cualquier punto de su programa.

Todos los campos de entrada son expresiones C.

2.3.10.3 View → Graph → Eye Diagram.

Este gráfico es usado para examinar cualitativamente la fidelidad de una señal. Por su parte, la señal de entrada es continuamente superpuesta una sobre la otra, dentro de un rango específico.

Para ver el gráfico del tipo Eye Diagram, seleccione desde el menú principal los comandos **View** ® **Graph** ® **Eye**, los que abrirán la ventana de diálogo "Graph Property". En ella podrá visualizar una columna izquierda que contiene el nombre de cada campo, y una derecha en la que podrá ajustar los valores de cada uno de estos campos; teniendo con ello la posibilidad de ajustar los valores que le sean necesarios. Una vez hecho esto basta con clicar "OK" y la ventana de gráfico aparecerá con los valores escogidos. En caso de que el usuario desee cambiar alguno de estos valores deberá clicar sobre la ventana de gráfico con el botón derecho del mouse, seleccionar "Properties", y ajustar los parámetros que desee. También puede actualizar el gráfico en cualquier punto del programa.

Todos los campos de entrada son expresiones C.

2.3.10.4 View → Graph → Image.

Este gráfico es usado para examinar los algoritmos de procesamiento de imagen, basando sus datos en RGB y YUV. Para visualizar las opciones del gráfico "Image", seleccione los comandos **View** ® **Graph** ® **Image**, del menú principal, los que presentarán la ventana de diálogo "Graph Property"; en la que podrá ver una columna izquierda que contiene el nombre de cada campo, y una derecha en la que podrá ajustar los valores de cada uno de estos campos; teniendo con ello la posibilidad de ajustar los valores que le sean necesarios. Una vez hecho esto basta con clicar "OK" y la ventana de gráfico aparecerá con los valores escogidos. En caso de que el usuario desee cambiar alguno de estos valores deberá clicar sobre la ventana de gráfico con el botón derecho del mouse, seleccionar "Properties", y ajustar los parámetros que desee. También puede actualizar el gráfico en cualquier punto del programa.

Todos los campos de entrada son expresiones C.

2.3.11 View →Tasks.

2.3.11.1 View →Tasks → Refresh Task.

Este comando no es soportado en la presente edición de Code Composer.

2.3.11.2 View → Tasks → (Process/thread) Not available.

Este comando no es soportado la presente edición de Code Composer.

2.3.12 View → Watch Window.

La ventana Watch le permite al usuario examinar y editar, las variables y expresiones de C, así como también expandir o comprimir expresiones complejas, además de evaluar términos y desplegar resultados en distintos formatos.

La característica principal de Quick Watch, es que permite agregar variables en forma rápida a la ventana Watch.

2.3.13 View → Call Stack.

La ventana Call Stack se utiliza para analizar la función de llamado que dirige la ubicación actual en el programa que se está depurando.

Para poder activar esta ventana, seleccione del menú principal: **View** ® **Call Stack** o cliquee el botón "View Stack" de la barra de herramientas de Debug.

Botón "View Stack": 

Para realizar el llamado de una función basta con desplegar el código fuente, y hacer doble click sobre la función que se desea. Luego aparecerá la ventana editora con el código fuente, y el cursor puesto en la línea donde se encuentra la función deseada. Así al seleccionar una función en la ventana "Call Stack", también se pueden observar las variables locales que se encuentran dentro del alcance de la función.

Las funciones llamadas están determinadas por el avance a través de la lista de enlaces que realizan los punteros de marco durante el tiempo que el stack se encuentre activo. Además el programa debe contar con una sección de stack predeterminada y una función principal, ya que sino, la ventana "Call Stack" desplegará el siguiente mensaje: "C source is not available".

La opción de call stack sólo trabaja con programas hechos en C.

2.3.14 View → Expression List.

Todas las expresiones y funciones de GEL son evaluadas con el "evaluador de expresiones". Para ir a esta opción, seleccione **View** ® **Expression List** del menú principal, lo que abrirá la

ventana de diálogo "Expressions Executing". Ésta, permite ver las expresiones que en esos momentos están siendo evaluadas por el evaluador de expresiones.

El usuario tiene también la opción de eliminar algunas de las expresiones que están siendo evaluadas, para lo cual debe seleccionar la expresión que desea eliminar y presionar el botón "Abort". Esto es muy útil cuando se está ejecutando una función de GEL que haya quedado "pegada en un loop", o está tomando un tiempo más de la cuenta en ejecutarse.

2.3.15 View → Project

La ventana de "Project View"; es desplegada automáticamente al iniciarse Code Composer y presenta gráficamente, el contenido del proyecto en uso.

En caso de que esta ventana no esté visible, el usuario deberá escoger del menú principal la opción **View** ® **Project**, para verla. Debajo de esta ventana se encuentran dos iconos: uno correspondiente al icono File y el otro al de Bookmarks, en caso de que desee seleccionar este último basta con clickearlo y la ventana correspondiente aparecerá..

Por otra parte, muchas de las operaciones del proyecto pueden ejecutarse desde la ventana Project View, para lo cual basta con usar el siguiente menú contextual:

- Para abrir un proyecto existente, haga click sobre "Project" con el botón derecho del mouse, y seleccione "Open project".
- Para abrir el contenido de la carpeta "Project", cliquee sobre el signo "+" de la misma, y luego en el siguiente.
- Para agregar un archivo al proyecto en uso, cliquee con el botón derecho del mouse sobre el nombre del proyecto y escoja la opción "Add Files".
- Para remover un archivo del proyecto, haga click con el botón derecho del mouse sobre el archivo y seleccione "Remove from project".

- Para compilar un archivo, cliquee con el botón derecho sobre el archivo y elija "Compile file".
- Para construir un proyecto entero, haga click con el botón derecho del mouse sobre el nombre del proyecto y seleccione "Build".
- Para editar un archivo que se esté usando en el proyecto, con el botón derecho del mouse haga click sobre el nombre del archivo y seleccione "Open".
- Para cerrar el proyecto en uso, con el botón derecho del mouse cliquee sobre el nombre del proyecto y escoja "Close".

2.3.16 View → Mixed Source/Asm

Además de ver las instrucciones desmontadas en la ventana de Dis-Assembly, el depurador de Code Composer le permite ver su código fuente en C intercalado con el código ya desamplado.

Para ver el código fuente en C y Assembler mezclados, después de haber cargado un programa en el simulador o en la tarjeta del DSP:

- 1) Seleccione el comando **View** ® **Mixed Source/ASM** del menú principal. (Al lado de él aparecerá una marca que indica que esta opción ha sido seleccionada).
- 2) Luego seleccione el comando **Debug** ® **Go Main**, también desde el menú principal.

El depurador comienza a ejecutar el programa y detiene la ejecución en "main ()" (que se encuentra al inicio del programa). El archivo fuente C asociado es desplegado automáticamente en la ventana editora. Note que las instrucciones desmontadas para cada expresión de C, aparecen dentro del código fuente, y tal como en la ventana de Dis-Assembly, la ubicación del PC es resaltada en color amarillo.

Usted puede escoger ver el código fuente en C con o sin el código assembler. En caso de querer cambiar la selección, escoja **View** ® **Mixed Source/ASM** del menú principal o cliquee

con el botón derecho del mouse sobre la ventana del código fuente y seleccione del menú contextual "Mixed Mode o Source Mode", según sea el caso.

2.3.17 View → Realtime Refresh Options.

Este comando no es soportado en la presente edición de Code Composer.

2.4 "Projects"

En la siguiente sección se detallarán las funciones de cada uno de los comandos de "PROJECT", la cuarta función del menú principal.

2.4.1 Project → New.

Sirve para crear un nuevo proyecto, y los pasos a seguir para ello son los siguientes:

- 1) Seleccione **Project** ® **New**, del menú principal. La ventana de diálogo "Save New Project As" se abrirá. Una vez ahí basta con escoger el directorio donde se desea guardar el nuevo proyecto que se creará. En caso de estar en el directorio equivocado, navegue hasta encontrar el correcto. Una vez escogido, se usará también para guardar los archivos restantes del proyecto, como por ejemplo, el archivo objeto que se genera por el compilado y ensamblado del proyecto. Es muy conveniente tener en cuenta el usar directorios diferentes cada vez que se cree un nuevo proyecto, ya que así , los archivos objetos de cada uno se guardan en sus respectivos directorios y además se les hace posible asignar diferentes opciones de compilación, assembler y linkado para cada proyecto.
- 2) En el campo "Nombre de Archivo", escriba el nombre que le asignará al nuevo proyecto, y luego cliquee "Guardar". Se creará un archivo de extensión **.mak**, con el

nombre asignado al nuevo proyecto, el que será contenido en la carpeta "Project", con sus respectivas subcarpetas. Ahora bien, si se abre un proyecto existente, las opciones de compilación, ensamblado y linkado son copiadas al nuevo proyecto, mientras que el proyecto existente es automáticamente cerrado. (Note que el título de banda de Code Composer cambia, desplegando ahora el nombre del Nuevo Proyecto).

- 3) Finalmente agregue sus archivos a la lista de proyecto a través de la opción **Project ® Add Files to Project**.

2.4.2 Project → Open.

Para abrir un proyecto ya creado, siga las acciones que a continuación se presentan:

- 1) Seleccione **Project ® Open**, desde el menú principal. La ventana de diálogo "Project Open" se abrirá. Luego seleccione el directorio que desea y si no lo encuentra navegue hasta encontrarlo.
- 2) Una vez escogido el directorio presione "Abrir", por lo que si tenía otro proyecto abierto, es automáticamente cerrado. Una vez cargado el archivo el título de banda de Code Composer cambia, desplegando ahora el nombre del proyecto abierto. En caso de que el archivo no sea cargado, aparecerá en pantalla un "mensaje de error" indicando que no se puede abrir el archivo escogido. En ese caso, verifique que haya seleccionado el archivo correcto. Si aún así indica que el archivo está dañado o por cualquier otro motivo no se puede abrir, entonces deberá crear un nuevo proyecto desde el inicio. Por otra parte, si desea abrir un archivo creado en una versión anterior de Code Composer, obtendrá un mensaje de advertencia que dice que el proyecto está en un formato antiguo. En tal caso presione OK para convertir el archivo al nuevo formato sin perder ningún tipo de dato. Si no se realiza la conversión del archivo éste no podrá ser abierto.

2.4.3 Project → Add Files to Project.

El "manejador del proyecto" (ver Anexo A) identifica a los archivos según la extensión de ellos. Para dar una idea sobre esto en la siguiente lista se presentan las extensiones utilizadas en Code Composer; y los archivos a los cuales corresponden cada una de ellas, suponiendo que fueron guardados en el lenguaje correcto con su correspondiente extensión.

<u>Extensión.</u>	<u>Tipo de archivo correspondiente.</u>
. o .c*	Archivo fuente en lenguaje C. Es aquél que el manejador del proyecto usará para su posterior compilación y linkado.
.a* o .s*	Archivo fuente en lenguaje Assembler. Al igual que el anterior, es aquél que el manejador del proyecto usará para su posterior compilación y linkado. Esto debido a que un archivo fuente dentro de Code Composer, puede ser realizado tanto en lenguaje C como en Assembler.
.o* o .lib	Archivo Objeto o de Librería. Es el que se utiliza "en" el linkado del proyecto.
.cmd	Archivo del comando Linker. Este archivo se utiliza "para" el linkado. (Mayor información sobre el archivo Linker, ir a Help ® Code Generation Tool Help , del menú principal).
otro	Archivo imposible de reconocer . El manejador del proyecto no le permitirá agregar otro tipo de archivos al proyecto.

Nota: No trate de especificar directamente archivos "include" o de cabecera, ya que son automáticamente agregados al proyecto, por el escaneo de archivos fuentes, que se realiza para ver las dependencias.

En lo que al archivo de comando Linker se refiere , "sólo uno" puede ser especificado para un proyecto, no en cambio el número de archivos que puedan ser sumados a él.

Todos los archivos sumados al proyecto son desplegados indicando su ruta completa, que es determinada cada vez que se abre un proyecto. Sin embargo al momento de ser almacenados son guardados con sólo una parte de la ruta, de tal manera, que el proyecto pueda ser fácilmente movido a un directorio diferente. Así, cada vez que se abre un proyecto, se define el nombre completo de la ruta y cuando se crea un archivo, se almacena sólo con un nombre relativo de esta ruta. Por ejemplo, si usted crea un archivo en la ruta: `c:\version1\linker\make\` y su archivo fuente está en la ruta `c:\version1\source\`, entonces la ruta relativa para su archivo fuente es: `..\..\source\`, donde cada uno de los dos puntos y la barra diagonal: `..\`, indican el respaldo de un nivel más arriba del directorio.

Ahora bien, si su archivo fuente es almacenado en otra dirección, entonces es almacenado con el nombre completo de la ruta, puesto que no existe una ruta relativa. En este caso si usted mueve o copia su archivo de proyecto creado desde `c:\version1\linker\make\` a `c:\version2\linker\make\`, Code Composer **asume** que el archivo fuente está en `c:\version2\source` al momento de abrir su proyecto. Por lo que es conveniente que cuando mueva un archivo creado, reescanee todas las dependencias para así asegurarse que Code Composer ha analizado cada una de las referencias.

2.4.3.a Como agregarle archivos al proyecto.

- 1) Del menú principal seleccione **Project** ® **Add Files to Project**, con lo que abrirá la ventana de diálogo "Add Files to Project ".

O bien :

Seleccione **View** ® **Project**, del menú principal para abrir la ventana "Project View", y después cliquee con el botón derecho del mouse sobre el nombre del proyecto y seleccione "Add Files".

- 2) En la ventana de diálogo de "Add Files to Project", especifique el archivo que desea añadir. Si el archivo no está en el directorio que se presenta, navegue buscando el directorio apropiado. Use la lista del campo "Tipos de Archivo" para escoger el tipo de archivo que busca.
- 3) Cliquee en "Abrir" para sumar a su proyecto el archivo especificado.

La ventana "Project View" despliega el contenido del proyecto y es actualizada cada vez que usted agrega un archivo. Si desea expandir la lista de Project, cliquee el signo "+", próximo a Project. Con esto verá todos los archivos que contiene su proyecto, los que son agrupados y separados en las siguientes carpetas:

<u>Carpeta.</u>	<u>Archivos que contiene.</u>
Include.	contiene todos los archivos de cabecera y los de tipo include: *.h.
Libraries.	contiene todos los archivos de librería: *.lib
Source.	contiene todos los archivos fuentes: *.c, *.asm

El archivo de comando Linker (*.cmd) aparece directamente bajo el archivo del proyecto (*.mak).

2.4.3.a Para remover un archivo del proyecto.

Para remover un archivo de su proyecto actual:

- 1) Seleccione **View** ® **Project**.
- 2) Luego cliquee con el botón derecho del mouse sobre el archivo que desea remover.
- 3) Finalmente seleccione "Remove from Project", del menú contextual.

2.4.4 Project → Close.

Para cerrar un proyecto se puede utilizar cualquiera de los siguientes pasos:

- Seleccione **Project** ® **Close**, del menú principal.
- Cree un nuevo proyecto.
- Abra otro proyecto.

2.4.4.a Usando la ventana Project View.

Para desplegar la ventana Project View, seleccione del menú principal el comando: **View** ® **Project**. Éste despliega la ventana "Project View", donde podrá ver el proyecto entero. Si hace click con el botón derecho del mouse sobre el **nombre del proyecto**, Code Composer le da posibilidad de cambiar cualquiera de las opciones referidas a él, con tan sólo seleccionar la opción apropiada del menú contextual.

2.4.5 Project → Compile File.

Seleccione **Project** ® **Compile File** del menú principal para compilar, sólo el archivo fuente actual, o bien utilice el botón de la barra de herramientas de "Project". Este comando no le permite linkar el archivo.

Botón "Compile": 

2.4.6 Project → Build.

Seleccione **Project** ® **Build** del menú principal, para construir el proyecto actual que está trabajando. Con esto, se compilan sólo los archivos que han cambiado desde la última construcción. Code Composer determina si un archivo debe ser compilado comparando el tiempo

de un archivo fuente con el de un archivo objeto. Si el tiempo del archivo fuente es más grande que el del archivo objeto, entonces el archivo es re-compilado.

Ahora bien para determinar si el archivo ejecutable debe ser re-linkado, Code Composer compara los tiempos del archivo objeto con el del ejecutable y si el del archivo objeto es más grande entonces lo vuelve a linkar.

2.4.7 Project → Rebuild All.

Seleccione **Project** ® **Rebuild All** del menú principal, para recompilar todos los archivos del proyecto que está trabajando y volver a linkar el ejecutable, en otras palabras reconstruye el proyecto.

2.4.8 Project → Stop Build.

Para detener el proceso de construcción del proyecto seleccione **Project** ® **Stop Build**, del menú principal, el que se detendrá sólo, después de finalizar el compilado del archivo en uso en esos momentos.

2.4.9 Project → Show Dependencies y 4.10 Project → Scan All Dependencies.

Durante el proceso de compilación, para determinar qué archivo debe ser compilado, el proyecto debe mantener una lista con las dependencias de los "include", para cada archivo. Así; Code Composer crea una dependencia que obliga, cuantas veces sea necesario ha reconstruir un proyecto y lo hace escaneando reiteradas veces todos los archivos fuentes que se encuentren en la lista de proyecto, los que corresponden a las directivas de #include, .include, y .copy. Además agrega cada uno de los archivos incluidos en la lista del proyecto, así como los archivos de "include", los que **no deben ser** agregados por el mismo usuario, ya que son agregados en forma automática por el manejador de proyecto. Éste busca los archivos "include" o de cabecera,

basándose en el tipo de archivo fuente, siendo el directorio actual, la ruta completa de este archivo, y la ruta relativa, la resuelta con respecto a este directorio.

La búsqueda es ejecutada en el siguiente orden:

Para archivos fuentes en C:

- Primero busca en el directorio actual.
- Luego, en la opción de compilado, dentro de la lista de rutas de include, de izquierda a derecha.
- En la lista de las rutas de include especificando el C_DIR de DOS, de izquierda a derecha.

Para archivos fuentes en Assembler:

- En el directorio actual.
- La opción de ensamblado, dentro de la lista de rutas de include, de izquierda a derecha.
- En la lista de las rutas de include especificado en el C_DIR de DOS, de izquierda a derecha.

Por su parte; Code Composer minimiza el tiempo de saneo de dependencias, ya que sólo escanea los nuevos archivos o los archivos que hayan sido modificados desde el último escaneo. Éstos son detectados por cualquier cambio que haya ocurrido tanto en los datos como en el tiempo de un archivo, incluyendo archivos de versiones anteriores que hayan sido reemplazados.

2.4.9.a Cómo ver las dependencias de un archivo.

Para generar las dependencias de un archivo, se pueden usar cualquiera de los siguientes métodos:

- Seleccione **Project** ® **Show Dependencies**, del menú principal. Con esto ejecuta un escaneo de cada dependencia, antes de desplegarla en el proyecto.
- Seleccione **Project** ® **Build**, del menú principal. Con ello se ejecuta un escaneo de dependencias, antes de construir un proyecto.
- Seleccione **Project** ® **Scan All Dependencies**, del menú principal o bien cliquee sobre el nombre del proyecto en la ventana View Project con el botón derecho del mouse, de manera que se escanearán todas las dependencias de todos los archivos, independientemente si éstos han cambiado o no desde el último escaneo de dependencias que se haya realizado.

2.4.9.b Como desplegar las dependencias del archivo.

- 1) Antes que nada, el proyecto debe estar abierto; si no lo está, seleccione **Project** ® **Open**, desde el menú principal. A continuación se abrirá la ventana de diálogo "Project Open".
- 2) En la ventana principal, seleccione el nombre del archivo de proyecto que desea abrir, en caso de que no se encuentre en el directorio que se presenta, navegue a través del explorador hasta encontrar el archivo que desea.
- 3) Luego seleccione **Project** ® **Show Dependencies**, del menú principal para ejecutar un escaneo de las dependencias y así asegurarse que sean actualizadas.

Cada vez que realice un escaneo de Dependencias, se abrirá la ventana "Dependencies" en la cual se podrán ver los archivos utilizados. En caso que alguno de los archivos del listado se encuentre en rojo, quiere decir que este archivo en particular no ha sido analizado por Code Composer y será reconstruido cuando sea llamada la opción: "Build", en el proyecto. Para cancelar el escaneo presione el botón "Cancel".

2.4.9.c Como excluir un archivo del escaneo de dependencias.

Para evitar el escaneo de dependencias en ciertos archivos, Code Composer usa un archivo de exclusión denominado: "exclude.dat", el que contiene una lista de los archivos "include" del sistema que improbablemente puedan cambiar. Así el usuario puede editar este archivo, ya sea, para excluir otros archivos del escaneo, tales como archivos de cabecera que nunca cambian o para incluir archivos del sistema que desee cambiar.

2.4.11 Project → Options.

Code Composer además trae todas las opciones de Compilación, Ensamblado y Linkado que necesita el proyecto para construir el programa. Ahora bien; para especificar cada una de estas opciones se deben seguir los siguientes pasos:

- 1) Seleccione **Project** ® **Options** del menú principal.

O bien

Cliquee con el sobre el nombre del proyecto en la ventana Project View con el botón derecho del mouse, y seleccione Options, del menú contextual.

- 2) Luego seleccione la etiqueta apropiada: Compiler, Assembler, o Linker.
- 3) De ellas seleccione las opciones que usará cuando construya su programa.
- 4) Finalmente cliquee OK para aceptar lo seleccionado.

2.4.3 Project → Recent project files.

Para recargar un archivo de proyecto usado recientemente, vaya al menú principal y seleccione en la lista de **Project** ® **Recent project files**, el nombre del archivo que desee cargar.

2.5 "Debug"

En la siguiente sección se detallarán las funciones de cada uno de los comandos de "DEBUG", la quinta función del menú principal.

2.5.1 Debug → Breakpoint.

Un Breakpoint, es un punto en el cual se detiene la ejecución del programa, así una vez detenido el usuario puede revisar el estado de éste, modificar o examinar variables, o revisar el call stack. Ahora bien para colocar un breakpoint, puede usar el botón de la barra de herramientas de "Project": Toggle Breakpoint o bien seleccionar desde el menú principal la opción **Debug** ® **Breakpoints**. Esto abrirá la ventana de diálogo "Break/Probe/Profile Points". Una vez que se ha puesto un breakpoint, el usuario tiene la posibilidad tanto de habilitarlo como de deshabilitarlo.

Nota: A continuación información para evitar un posible daño en el procesador:

- No ponga un breakpoint como llamado de retardo sobre una instrucción ejecutada.
- No ponga un breakpoint sobre la última o las dos últimas instrucciones antes del final de un bloque de instrucción de repetición .

2.5.2 Debug → Probe Point.

Un Probe Point, es un punto en el cual se permite actualizar una ventana en particular o leer y escribir muestras desde un archivo que tiene lugar en un punto específico de su algoritmo. Éste

conecta una señal de prueba para ese punto en su algoritmo. Una vez conectados usted puede habilitar o deshabilitar los puntos de prueba, igual que los breakpoints.

Una ventana creada, por defecto, es actualizada en cada breakpoint. Sin embargo, el usuario puede cambiar esto, de manera que la ventana sea actualizada sólo cuando el programa alcance los Probe Points conectados. La ejecución del programa continúa una vez actualizada la ventana.

Junto con poder usar Probe Points en los archivos I/O de Code Composer, también puede usarlos para conectar señales de dato en un punto particular del código del DSP. Cuando un Probe Point es alcanzado dentro del algoritmo, los datos son echados a correr desde un área específica de la memoria a un archivo o desde un archivo a la memoria.

Nota: Cuando se encuentra un Probe Point el procesador de la tarjeta es temporalmente detenido por el procesador del Pc, por lo tanto la aplicación de la tarjeta puede no ser capaz de encontrar la restricción de un "Probe Point" en tiempo real.

A continuación para dar "single steps" (ver Anexo A) dentro del código, use los siguientes botones de la barra de herramientas de Debug.

2.5.3 Debug → SingleInto.

El comando Step Into sigue los rastros detrás de la ejecución de las instrucciones del programa. Para realizar un Step Into a través del código, basta con hacer click sobre el botón de la barra de herramientas de Debug o bien seleccionar **Debug** ® **StepInto** del menú principal. Ahora, si está trabajando sobre un archivo fuente programado en C o en Assembler, este comando avanza de una en una por las instrucciones de C o Assembler, según corresponda.

2.5.4 Debug → StepOver.

El comando Step Over es usado para avanzar paso a paso a través de la función actual y para ejecutar órdenes individuales en ella. Esto es, si una función llamada es encontrada, la función se ejecuta por completo pero si se encuentra con un breakpoint, primero se detiene y después llama a la función. Para poner un step over, basta con seleccionar **Debug** ® **StepOver**, desde el menú principal, o bien haciendo click sobre el botón "step over" de la barra de herramientas de Debug.

2.5.5 Debug → StepOut.

Si se encuentra dentro de una subrutina, puede seleccionar el comando Step Out para completar la ejecución de ella. Así la ejecución se detiene, después que la subrutina actualmente corriendo retorna al llamado de la función. Esto lo puede realizar cada vez que cliquee sobre el botón "Step Out" de la barra de herramientas de Debug o bien seleccionando **Debug** ® **StepOut**, del menú principal.

2.5.6 Debug → Run.

Este comando le permite ejecutar un programa. Para ello basta con seleccionar desde el menú principal la opción **Debug** ® **Run** o bien hacer click en el botón "Run" de la barra de herramientas de Debug. (En caso que se haya puesto algún breakpoint, la ejecución continúa hasta que éste es encontrado).

2.5.7 Debug → Halt.

Para detener la ejecución de un programa basta con seleccionar el botón Halt de la barra de herramientas de Debug o seleccionar **Debug** ® **Halt** desde el menú principal.

2.5.8 Debug → Animate

Con esta opción el usuario puede "animar" su programa, es decir, agregarle diversos puntos de análisis (Probe Point, Profile Point, Breakpoint) a éste mientras está corriendo, para ver y analizar su ejecución. Para esto cliquee sobre el botón "Animate" o bien seleccione **Debug** ® **Animate**, del menú principal. El programa correrá hasta encontrar un breakpoint, luego actualizará la ventana que no esté conectada a Probe Points y reanudará la ejecución. Para detener la animación seleccione **Debug** ® **Halt**, del menú principal. En caso que desee tener control de la animación seleccione **Option** ® **Animate Speed**, del menú principal.

2.5.9 Debug → Run Free.

Este comando, como su nombre lo indica, deja correr el programa libremente, es decir, deshabilita todos los Breakpoints, Probe Points y los Profile Points antes de la ejecución de un programa, a partir de la ubicación actual del PC (contador de programa activo). Para esto seleccione **Debug** ® **Run Free** del menú principal. Para detener la ejecución escoja el comando **Debug** ® **Halt**, del menú principal

Nota: El simulador no soporta el comando "Run free".

2.5.10 Debug → Run to Cursor.

Sirve para hacer correr un programa cargado hasta que encuentre la posición del cursor en la ventana de Dis-Assembly. Para escoger este comando diríjase al menú principal y escoja **Debug** ® **Run to Cursor**.

2.5.11 Debug → Multiple Operation.

Sirve para llamar a diversos Singles Step en un solo paso (también llamado "Stepping Command Multiple Times"). Para ello:

- 1) Seleccione el comando **Debug** ® **Multiple Operations**, del menú principal. Se abrirá la ventana de diálogo "Multiple Operations".
- 2) Seleccione uno de los comandos "Stepping" de la lista que se presenta.
- 3) Especifique el número de veces que desea que el comando sea activado.
- 4) Cliquee "OK".

Repita este procedimiento para activar el mismo u otro Single Step.

2.5.12 Debug → Reset DSP.

Para resetear el procesador de la tarjeta, se encuentra disponible el siguiente comando:

Reset DSP: el comando **Debug** ® **Reset DSP** inicializa el contenido de todos los registros a su estado power-up de acuerdo a las especificaciones de simulación del DSP y detiene la ejecución de su programa. En caso que la tarjeta no responda a este comando, y se esté usando un driver (de la tarjeta) basado en un kernel, puede ser que el kernel esté dañado, así es que deberá ser recargado.

2.5.13 Debug → Restart.

El comando **Debug** ® **Restart** restablece el PC al punto de partida para el archivo cargado actualmente.

2.5.14 Debug → Go Main.

El comando **Debug** ® **Go Main** sitúa temporalmente un breakpoint en "main()" (dentro del programa), y luego inicia la ejecución. Así este breakpoint es quitado, una vez que éste es

encontrado o cuando la ejecución es detenida. Esta es una buena forma para iniciar aplicaciones en programadores de C, ya que así el archivo asociado es automáticamente cargado, cuando la ejecución se detiene en "main()".

2.5.15 Debug → Enable Task Level Debugging.

Este comando no es soportado en la realización de Code Composer.

2.5.16 Debug → Real Time Mode.

Este comando no es soportado en la realización de Code Composer.

2.5.17 Debug → Enable Urde Real Time Mode.

Este comando no es soportado en la realización de Code Composer.

2.6 "Profiler"

En la siguiente sección se detallarán las funciones de cada uno de los comandos de "PROFILER", la sexta función del menú principal.

2.6.1 Profiler → Profile Points.

Un Profile Point son breakpoints especiales que capturan información en lugares específicos del programa. Cada profile points cuenta el número de veces que fue disparado, guardando estadísticas sobre el número de ciclos o de otros eventos transcurridos desde el anterior profile point que fue disparado. A diferencia de los breakpoint, los profile points después de guardar una estadística de la ejecución, hacen un resumen de ella.

Ahora bien, para que los profile points puedan llevar una estadística de los ciclos de instrucción o de otros eventos, el clock o reloj debe encontrarse habilitado, en caso contrario profile point sólo es capaz de contar el número de eventos de cada profile point, sin ser capaz de generar otras estadísticas. Una vez puesto un profile point el usuario puede habilitarlo o deshabilitarlo.

2.6.1.a Como poner un Profile Points.

El usuario puede crear un Profile Points poniendo el cursor sobre la línea del archivo fuente en que desee colocarlo o bien en la ventana "Dis-assembly" en el lugar donde desee que se encuentre. Luego basta clicar el botón "Profile Point" de la barra de herramientas.

Botón "Profile Point": 

2.6.1.b Como remover un Profile Point.

- 1) Desde el menú, seleccione **Profilers** ® **Profilers Points**. Se abrirá la ventana de diálogo "Break/Probe/Profile Points".
- 2) En la etiqueta "Profile Point", seleccione un profile point.
- 3) Luego presione el botón "Delete".
- 4) Presione "Aceptar" para cerrar la ventana de diálogo.

2.6.1.c Borrar todos los Profiler Points.

Presione el botón de la barra de herramientas de Project:

Botón "Remove All Profile Points": 

O bien:

Puede eliminar los Profiler Points, usando los comandos del menú principal. Para ello:

- 1) Del menú seleccione **Profiler** ® **Profile Points**. La ventana de diálogo Break/Probe/Profile Points aparecerá.
- 2) Presione el botón "Delete All".
- 3) Presione "Aceptar" para cerrar la ventana de diálogo.

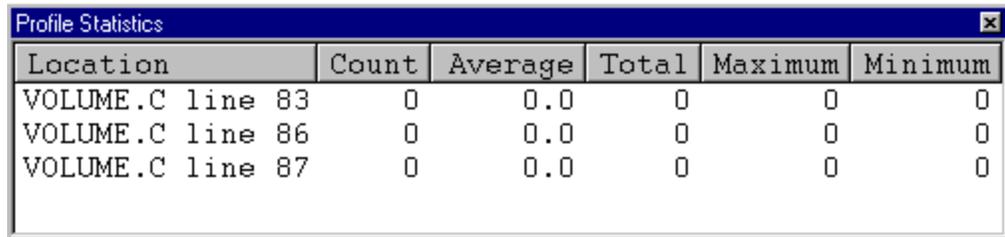
2.6.2 Profiler → View Statistics.

Para ver las estadísticas de profiler, seleccione **Profiler** ® **View Statistics** del menú principal. Esto abrirá la ventana "Profile Statistics". Esta ventana despliega los resultados de cada profile point. Cada punto despliega el número de veces que ha sido disparado y las estadísticas recogidas, ya sean sobre el número de ciclos u otros eventos que hayan transcurridos desde el último profile point disparado. Además las estadísticas incluyen los máximos y mínimos, el total y el promedio del número de ciclos.

La ventana de Profile Statistics se actualiza cada vez que es disparado un profile point, pero tener muchas ventanas actualizadas pueden bajarle la velocidad a la ejecución del profiling. Para que no suceda esto, hay dos maneras de reducir el número de veces de actualización de la ventana: una es conectando la ventana a un Probe Point, o abrir y cerrar la ventana tantas veces como sea necesario. Así, una ventana "Profile Statistics" conectada a un Probe Point, sólo se actualiza cuando un Probe Point es disparado, lo que le permite tener más control sobre ella. Esta ventana, además, sólo despliega los resultados de Profiler y no necesita acumular estadísticas de profiler, por lo cual no es necesario que esté abierta todo el tiempo.

2.6.2.a Cómo limpiar las estadísticas de un "Profile Point".

- 1) Seleccione el profile point que desea, en la ventana "Profiler Estatics", usando el mouse o los cursores del teclado. El profile point seleccionado será delineado con una línea punteada.



Location	Count	Average	Total	Maximum	Minimum
VOLUME.C line 83	0	0.0	0	0	0
VOLUME.C line 86	0	0.0	0	0	0
VOLUME.C line 87	0	0.0	0	0	0

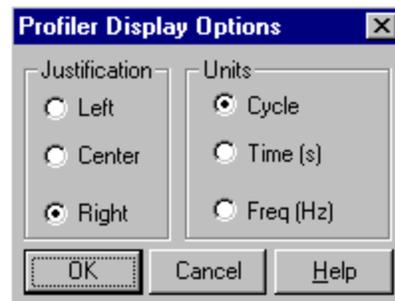
Luego presione el botón derecho del mouse y escoja "Clear". Con ello los campos de: cuenta, promedio, total, máximo y mínimo son puestos en 0. No existe cancelación para esta acción en el menú de Edit.

2.6.2.b Cómo limpiar las estadísticas para todos los Profile Point.

Presione el botón derecho del mouse y seleccione "Clear All". Todos los campos de: cuentas, promedios, totales, máximos y mínimos, serán llevadas a 0. No hay manera de deshacer esta acción a través de la opción Edit del menú.

2.6.2.c Cómo cambiar las opciones desplegadas.

- 1) Haga click desde el menú principal sobre Profile → View Statistics, para efectuar la actualización.
- 2) Presione el botón derecho del mouse y escoja **Propiedades** ® **Display Options**. Aparecerá la ventana de diálogo "Profiler Display Options".



En el campo Justificación, el usuario puede cambiar el despliegue de datos seleccionando cualquiera de los botones de elección ya sea, Izquierda, Centrado o Derecha.

Por otra parte, en el campo Units, puede cambiar la unidad de medida en la que aparecerán los datos de la ventana "Profile Statistics", seleccionando ya sea Ciclo, Tiempo, o Frecuencia. La unidad de medida puede ser la cuenta de ciclos, el tiempo medido en segundos, o la frecuencia medida en Hertz, pudiendo configurar el tiempo del ciclo de instrucción a través de la opción "Clock Setup".

Los tiempos de ciclos son convertidos a frecuencia tomando números alternados, esto, ya que las unidades de frecuencia suelen ser más útiles, por ejemplo, para determinar si su programa puede manejar una determinada frecuencia de muestreo.

Por su parte la cuenta de ciclos son convertidas al dominio del tiempo, multiplicando la cuenta del ciclo por el tiempo del ciclo de instrucción.

2.6.3 Profiler → Enable Clock.

Profile clock cuenta los ciclos de instrucción del procesador u otros eventos durante las operaciones run y single step.

Profile clock es accesible como una variable denominada CLK, o a través de la ventana "Clock". Además la variable CLK puede ser vista en la ventana "Watch" y modificada en la

ventana de diálogo "Edit Variable". También se puede acceder a CLK por medio de las funciones de GEL definidas por el usuario.

Los ciclos de instrucciones son medidos de diferente manera, dependiendo que drivers de DSP se está usando. Así, otros drivers pueden requerir otro tipo de conteo. Para manejar estos recursos debe habilitar y deshabilitar profile clock.

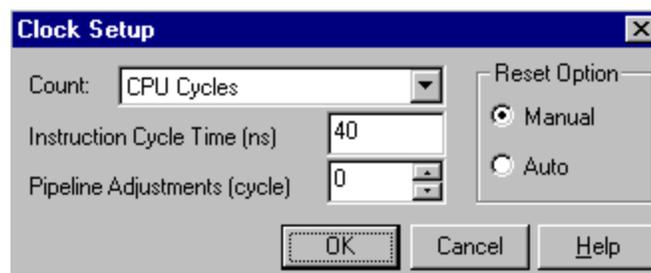
Cuando el reloj es habilitado el depurador de Code Composer toma más de los recursos necesarios, para implementar el conteo del ciclo de instrucción. Cuando éste se encuentra deshabilitado, las fuentes se encuentran disponibles para usted.

2.6.3.a Como Habilitar/Deshabilitar un Profile Clock.

Para habilitar o deshabilitar el reloj, seleccione desde el menú principal **Profiler** ® **Enable Clock**. Una marca será desplegada junto a este ítem del menú, cuando el reloj haya sido habilitado, en caso contrario, no es presentada.

2.6.4 Profiler → Clock Setup.

Para cambiar a Profiler Clock Setup (o la Setup de Profiler Clock), seleccione **Profiler** ® **Clock Setup** del menú principal, lo que abrirá la ventana de diálogo "Clock Setup".



En el campo de: "Instruction Cycle Time", usted puede ingresar el tiempo de ejecución de una instrucción. Esta información es usada para convertir el conteo de ciclos a los dominios de tiempo o de frecuencia para así desplegar las estadísticas de profiler.

De la lista que se presenta en el campo "Count", usted puede seleccionar el evento a perfilar. Ahora bien, "CPU cycles", puede estar solo en la lista de opciones, como también pueden haber otras opciones, lo que va a depender del driver que se está usando. Éstos pueden incluir: el número de interrupciones, el de subrutinas o retorno de interrupciones, el número de divisiones, de llamados de subrutinas; etc. Por ejemplo si usted selecciona "divisiones", la variable CLK, cuenta el número de divisiones, en lugar de contar los ciclos de la CPU.

Nota: El simulador sólo despliega los parámetros de "CPU Cycles", en el campo Count, ya que él no perfila otro evento del DSP a través de la interfaz con un simulador de la tarjeta del DSP.

Usted puede usar el parámetro "Reset Option", para determinar como será acumulada la variable CLK. Si usted selecciona la opción "Manual", la variable CLK acumula cuentas de los ciclos de instrucción, sin resetear el reloj. Por otra parte si selecciona la opción Auto, la variable CLK es reseteada automáticamente (puesta a 0), antes de echar a correr o de poner en marcha el procesador de la tarjeta, por lo que la variable CLK sólo despliega los ciclos de la última puesta en marcha.

Además de ello se encuentra el campo "Pipeline Adjustment", (que es el ajustamiento de datos cuando pasan por procesador central) para compensar el número de ciclos usados en la limpieza de una parte de la memoria del procesador, cuando se reparan los breakpoints. Cada vez que el procesador da un paso, se detiene a reparar los breakpoints y debe limpiar la parte de la memoria en la cual pasaron o existieron datos. Los ciclos necesarios para realizar esto, dependen del estado del número de programas esperando, el cual no es conocido. Para tener un conteo de ciclos más exactos se debe especificar un valor al ser sustraído del conteo del ciclo para compensarlo por este efecto.

2.6.5 Profiler → View Clock.

Como ya se describía con anterioridad a profile clock se le es accesible ya sea a través de la variable denominada CLK, como por la ventana de Clock, donde la variable CLK puede ser vista en la ventana "Watch" y modificada en la ventana de diálogo "Edit Variable", haciendo click en el menú principal en la opción **Edit** ® **Edit Variable**. CLK también está disponible para ser definida por el usuario en las funciones de GEL.

2.6.5.a Cómo ver un Profile Clock.

Seleccione **Profiler** ® **View Clock** del menú principal. Aparecerá la ventana "Clock" presentando el valor de la variable CLK.

2.6.5.b Cómo resetear un Profile Clock.

Para resetear un Profile Clock; edite la variable CLK estableciéndola en 0, o bien haga doble click, en los contenidos de la ventana "Clock".

2.7 "Option"

En la siguiente sección se detallarán las funciones de cada uno de los comandos de "OPTION", la séptima función del menú principal.

2.7.1 Option → Color.

Para cambiar el color de los elementos de la pantalla dentro de Code Composer, siga los pasos que a continuación se detallan:

- 1) Del menú principal, seleccione el comando **Option** ® **Color** para abrir la ventana de diálogo "Color".
- 2) Luego, de la lista del campo "Screen Element", seleccione el elemento de la pantalla que desea cambiar de color.
- 3) Después, cliquee el color deseado en "Palette".
- 4) Una vez hecho los cambios, cliquee el botón "OK".

2.7.2 Option → Font.

Para cambiar el tipo de fuente y/o sus características, que por defecto usa Code Composer, siga los siguientes pasos:

- 1) Seleccione el comando **Option** ® **Font**, del menú principal, lo que abrirá la ventana de diálogo "Fuente".
- 2) Seleccione una de las fuentes que allí se presentan.
- 3) Luego, del listado del campo "Estilo de Fuentes", seleccione un estilo para la fuente escogida, los que varían según la fuente.
- 4) Finalmente de la lista de "Tamaño", seleccione un tamaño para la fuente escogida. Los tamaños también varían según la fuente.

En el campo "Muestra" se visualiza la fuente escogida y sus características, tal como aparecerán en la interfaz de Code Composer.

Para restaurar la fuente original y sus características, seleccione:

- Tipo de Fuente: Courier.
- Estilo: Regular.
- Tamaño: 10.

2.7.3 Option → Editor Properties.

El usuario, también tiene la posibilidad de personalizar las opciones del editor que usa frecuentemente. Para ello realice las siguientes acciones:

2.7.3.a Cómo determinar las propiedades del Editor.

- 1) Seleccione **Option** ® **Editor**, del menú principal, lo que abrirá la ventana de diálogo "Editor Properties".
- 2) En esta ventana usted podrá escoger entre las siguientes opciones:
 - **Tab Stops**. En este campo ingrese el número que desee de tab stops, o en otras palabras el número de columnas que avanza el cursor cuando se presiona el tabulador. El valor presentado por defecto es: 4.
 - **Open files as read only**. Si desea prevenir las modificaciones involuntarias de un archivo, esta opción debe encontrarse habilitada al abrir las ventanas. Estando esta opción deshabilitada, el usuario puede hacer cualquier tipo de modificación a los archivos que desee abrir.
 - **Save before running tools**. El habilitar esta opción le incita al usuario a guardar sus archivos a través de la ventana de diálogo "Save Changes". Esto ocurre cada vez que se llama un proyecto construido, en el que se han realizado cambios en cualquiera de sus archivos actualmente abiertos.
 - **Recent files**: Seleccione el número de archivos recientemente usados que desea que aparezcan en los ítem de menú de: **File** ® **Recent Source Files**, o **Project** ® **Recent Project Files**.

3) Finalmente cliquee "OK".

2.7.4 Option → Keyboard.

Esta opción le permite al usuario asignarle a todos los comandos del menú IDE, su correspondiente combinación de teclas con la que podrá trabajar sin necesidad de ocupar el mouse. Para ello seleccione **Option** ® **Keyboard** del menú principal, lo que abrirá la ventana de diálogo "Cuztomize Keyboard" donde podrá asignar la respectiva combinación de teclas para cada comando.

Una vez en la ventana, en la caja de diálogo "Commands", seleccione el comando que desea personalizar, mientras que en la ventana "Assigned Keys" puede ver la actual combinación de teclas designadas para ese comando.

Para asignarle una nueva combinación de teclas; al comando seleccionado, presione el botón "Add", el que abrirá la ventana de diálogo "Assign Shortcut". En ella, ingrese la nueva secuencia de teclas y presione "OK".

Ahora bien si desea quitar una secuencia de teclas en particular para un comando, primero seleccione la secuencia determinada en la ventana "Assigned Keys" y luego presione el botón "Remove".

Finalmente, con la opción "Save As", el usuario podrá guardar la configuración del teclado en un archivo. Este botón traerá la ventana de diálogo "Guardar Como" donde podrá navegar al directorio donde desee guardar su configuración.

Para navegar y cargar el archivo de una configuración previamente guardada, utilice el botón "...".

2.7.5 Option → Animate Speed.

Con esta opción el usuario puede controlar la velocidad de la animación que es el tiempo mínimo entre breakpoints (punto en que el programa se detiene para permitir detectar y eliminar errores). En el modo de animación, el programa del DSP se ejecuta hasta que encuentra un breakpoint. En este punto, la ejecución se detiene y todas las ventanas que no estén conectadas a algún "Probe Point" se actualizan. Luego el programa en ejecución realiza un resume hasta el próximo breakpoint. (Un "Probe Point" siempre reanuda la ejecución después de actualizar la ventana vinculada a él).

2.7.5.a Como seleccionar la Velocidad de Animación.

- 1) Seleccione el comando **Option** ® **Animate Speed**. La ventana de diálogo de "animate Speed Properties" aparecerá.
- 2) En ella, ingrese la velocidad de animación, en segundos.
- 3) Por último presione el botón OK.

La ejecución del programa no es resumida hasta que el tiempo mínimo haya terminado desde el breakpoint anterior.

Para animar, seleccione **Debug** ® **Animate** del menú principal y para detener la animación, seleccione **Debug** ® **Halt** desde el menú principal.

2.7.6 Option → Memory Map.

Usted puede usar la ventana de diálogo "Memory Map" para definir y enumerar su mapa de memoria interactivamente. Para ello seleccione con el comando **Option** ® **Memory Map** en la barra de menú principal.

Cuando usted inicia Code Composer, el mapa de memoria está desactivado, pudiendo así, acceder a cualquier lugar de memoria; sin que el mapa de memoria interfiera.

2.7.6.a Cómo agregar un nuevo rango de Mapa de Memoria.

Use los pasos siguientes para definir el rango de memoria que al que desea acceder:

- 1) Seleccione el comando **Option** ® **Memory Map** del menú principal. Este traerá la ventana de diálogo "Memory Map".
- 2) Asegúrese que el casillero de "Enable Memory Mapping" esté activado. De lo contrario, toda la memoria (RAM) accesible para esa dirección, es asumida como válida por el depurador de Code Composer.
- 3) Seleccione la carpeta correspondiente a la página que quiere modificar (Programa, Datos, o I/O). Sáltese este paso si está usando un procesador que tiene sólo una página de memoria. Cuando el procesador tiene sólo una página de memoria, se crea sólo una carpeta.
- 4) Ingrese la dirección de inicio del nuevo rango de mapa de memoria en "Starting Address" del campo de datos.
- 5) Luego ingrese la longitud del nuevo rango en Length.
- 6) Seleccione las características del lectura/escritura del nuevo rango de memoria en el campo de Atributos de Archivo.
- 7) Finalmente presione el botón "Add".

De esta forma el depurador le permitirá ingresar un nuevo rango de memoria, el que es asumido como válido, en consecuencia, los atributos de este nuevo rango son también cambiados.

Así, el usuario puede cambiar los atributos de lectura/escritura de este nuevo rango. Esto lo puede hacer definiendo un nuevo mapa de memoria (con la misma dirección de inicio y longitud), para luego clicar el botón "Add". Con esto el depurador sobrescribe los nuevos atributos en los ya existentes.

2.7.6.b Como borrar un rango de mapa de memoria.

Usted puede borrar un rango de mapa de memoria, además de cambiar el campo "Attributes" a "None - No Memory/Protected", lo cual significa que no puede ni escribir ni leer en esta parte de la memoria. Para borrar un rango de mapa de memoria realice los siguientes pasos:

- 1) Del menú principal seleccione el comando **Option** ® **Memory Map**, lo que abrirá la ventana de diálogo "Memory Map".
- 2) Del cuadro "Memory Map List" (dentro de la ventana de Memory Map), seleccione el rango del mapa de memoria que desea borrar.
- 3) Finalmente cliquee el botón "Delete".

Así cuando se intenta leer desde una ubicación de la memoria, protegida por el mapa de memoria, el depurador reemplaza esta lectura por un valor protegido, en lugar de intentar leerlo desde la tarjeta. El valor inicial, por defecto, es 0; por lo tanto todas las ubicaciones de memoria nulas, presentan el valor 0. Sin embargo usted puede cambiar este valor que por defecto aquí es presentado, ingresando su propio valor en el campo de entrada "Protected Value" de la ventana de diálogo "Memory Map".

2.7.7 Option → Dis-Assembly Style.

Code Composer le ofrece diversas opciones para cambiar la forma de ver la información en la ventana Dis-Assembly. Así, en la ventana de diálogo "Dis-Assembly Style Options" podrá

escoger entre diversas opciones para ver su sesión de depurado. Por ejemplo, puede seleccionar, la dirección de base como decimal o hexadecimal.

2.7.7.a Como ver "Dis-Assembly Style Options".

- 1) Del menú principal escoja el comando **Option ® Dis-Assembly Style**.

O bien,

Cliquee con el botón derecho del mouse sobre la ventana Dis-Assembly y escoja **Properties ® Dis-Assembly**, del menú contextual.

- 2) Luego ingrese sus preferencias en la ventana de diálogo "Dis-Assembly Style Options".
- 3) Finalmente cliquee OK.

Nota: el contenido de la ventana Dis-Assembly es inmediatamente actualizado con el nuevo estilo.

2.7.8 Option → Program Loadw.

Esta opción presenta la ventana de diálogo de "Program Load Option", en la cual el usuario podrá seleccionar dos opciones con respecto a la carga del programa. Una de ellas corresponde a la verificación de la ejecución del programa después de cargarlo y la otra corresponde a si se desea cargar el programa después de construir. Usted podrá seleccionar cualquiera de estas opciones, con sólo marcarlas dentro de la ventana de diálogo.

2.7.9 Option → Connect Probe Points.

Para conectar un Probe Point a una ventana (activa), siga los pasos que a continuación se presentan:

- 1) En primer lugar abra la ventana a la que desea conectar un Probe Point.
- 2) Luego posicione el cursor sobre la línea en la cual desea colocar el probe Point, y cliquee el botón de la barra de herramientas: "Toggle Probe Point".

Botón "Toggle Probe Point": 

- 3) Después, seleccione **Debug** ® **Probe Points**, del menú principal, lo que abrirá la ventana de diálogo "Break/Probe/Profile Points", en ella escoja la etiqueta "Probe Points". En la ventana "Probe Point", aparecerá el nuevo "Probe Point" que haya creado, el cual le indicará que actualmente no hay conexión.

✔ echocan.c line 191 --> No Connection

Seleccione este punto para actualizarlo. Ahora podrá editar cada uno de los campos de esta ventana.

- 4) De la lista del campo "Probe type", seleccione un tipo de "Probe Point". Por defecto es incondicional. Lo que significa que cuando la ejecución del código alcanza un Probe Point, el archivo o ventana conectada es actualizada, continuando con la ejecución después de la actualización.
- 5) Luego, ingrese la ubicación en la cual desea poner un Probe Point, para ello siga cualquiera de los siguientes métodos, (aunque si se usó el botón "Toggle Probe Point" esté campo ya se encuentra con los valores apropiados):

- Para una dirección completa, puede ingresar el nombre de una función de C, o el nombre de una instrucción de Assembler.
 - O bien, ingrese la ubicación del Probe Point basándose de su archivo fuente en C, muy conveniente cuando no se sabe donde está la instrucción dentro del ejecutable. El formato para ingresar una ubicación basado en un archivo fuente en C es la siguiente:
Nombre_ de_ Archivo línea Número_de_línea.
- 6) Si en el momento de realizar el paso 4) ingresa un Probe Point condicional, entonces deberá ingresar la condición en el campo "Expression".
 - 7) Luego, conecte el Probe Point a un archivo o ventana. De lista "Connect To" que contiene todos los archivos y ventanas que pueden ser conectados a un Probe Point, elija el apropiado.
 - 8) Presione el botón "Add" para crear un nuevo Probe Point o presione "Replace", para modificar el actual Probe Point.

2.8 "Gel".

En la siguiente sección se detallarán las funciones de cada uno de los comandos de "GEL", la octava función del menú principal.

2.8.1 Agregando funciones Gel.

Usted puede agregar funciones de GEL a las que frecuentemente accede, a través del menú de GEL de la barra de menú de Code Composer. Para hacer esto, cree nuevas funciones de GEL, utilizando palabras claves para ellas, las que serán agregadas después en el menú de la lista de GEL.

2.8.2 Funciones de GEL predefinidas.

En esta realización de Code Composer no se contemplan funciones de Gel predefinidas.

2.9 "Tools".

En la siguiente sección se detallarán las funciones de cada uno de los comandos de "TOOLS", la novena función del menú principal.

2.9.1 Tools → List of tools.

En este menú, se ofrecen la opciones de:

Command Window. La habilitación de esta ventana le permite al usuario buscar ayuda acerca de los comandos usados en Code Composer.

Online Help. A través de esta opción el usuario podrá acceder a la ayuda en línea de cada una de las herramientas con sólo clicar en el nombre de una de ellas.

O bien:

Seleccione **Help** ® **General Help**, del menú principal, lo que abrirá la ventana "TMS320C3x Code Composer Help", desde la cual podrá acceder a los contenidos de la ayuda de cada una de las herramienta o bien podrá buscar la palabra que desee, a través del índice, para encontrar la información específica de ella.

2.10 "Window"

"WINDOWS", es la décima opción, en la que se presentan los comandos que determinan el tamaño de una ventana activa, como así el contenido de ella.

2.11 "Help"

Por último, se encuentra "HELP" el que contiene los distintos tipos de ayuda que le ofrece Code Composer, entre ellos un tutorial y la funciones específicas de cada opción y comando.

CAPITULO III:

"DISEÑO DE PROYECTOS CON CODE COMPOSER"

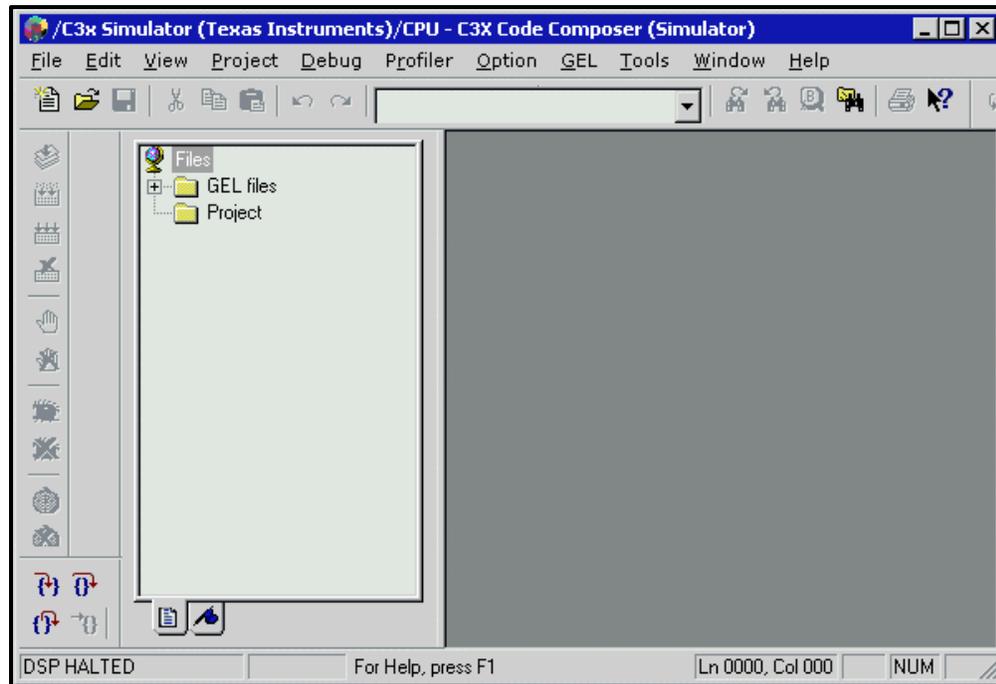
3.0 Introducción.

Como ya se explicaba con anterioridad, el propósito de este trabajo es realizar un CD multimedial que contenga el Manual de uso de Code Composer para posteriores trabajos prácticos basados en el uso de la tarjeta y con ella obviamente el de Code Composer; por lo que se pretende con ello, que un usuario cualquiera sea capaz de crear un determinado proyecto según las especificaciones que aquí se describen para poder hacer uso de este software y luego observar el resultado mediante un instrumental, a través del programa **DSK3D**.

Es así; que esta guía intenta presentar los conocimientos básicos necesarios para su realización, partiendo de un proyecto de ejemplo, hasta concluir con la observación de los resultados, por ejemplo, con el osciloscopio.

Para ello se explicará en detalle los pasos a seguir en el desarrollo o creación de un nuevo proyecto, dando a conocer; además el uso de iconos presentes en pantalla y las opciones que se despliegan en cada uno de los comandos de Menú.

La presentación de Code Composer es la siguiente:



A continuación se detallan los pasos a seguir en el uso de Code Composer, para el desarrollo o creación de cualquier aplicación.

3.1 Creación de una Aplicación.

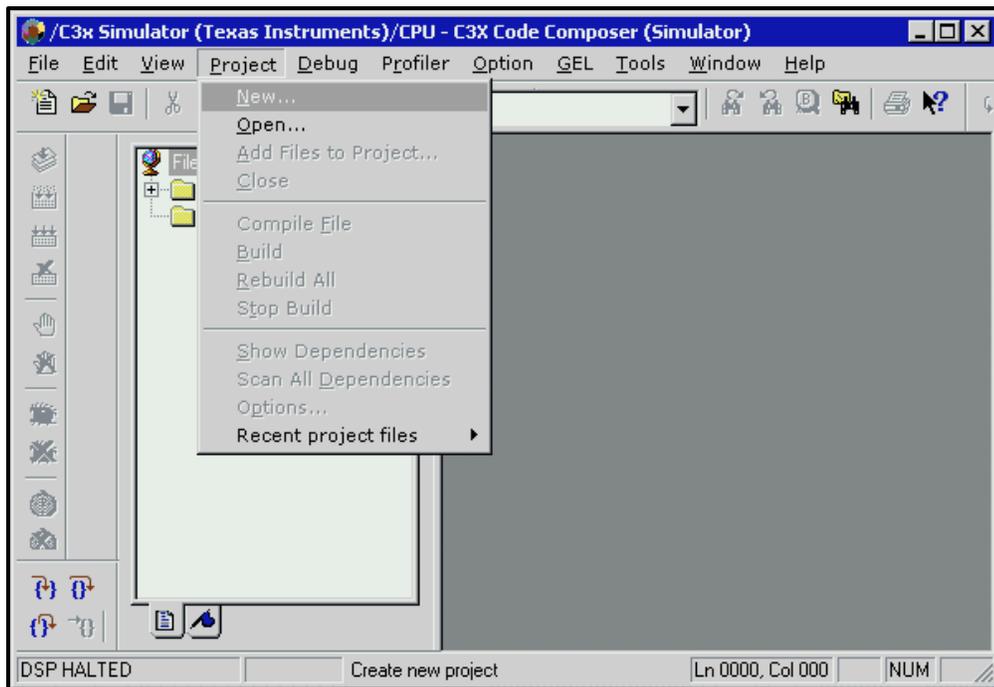
El primer paso a considerar en la creación de un proyecto es el lugar dentro de nuestro computador; donde el mismo, va a estar ubicado. Para ello es necesario crear una carpeta o directorio de trabajo donde se almacenarán los archivos necesarios, tanto intermedios como finales, del proyecto.

Por defecto **Code Composer Studio** comienza dentro de la carpeta my projects. Por su parte; el alumno puede crear un nuevo directorio de trabajo en el lugar que considere más apropiado. En este ejemplo se desea llamar a este directorio ubicado en el escritorio: "ejemplo_cc".

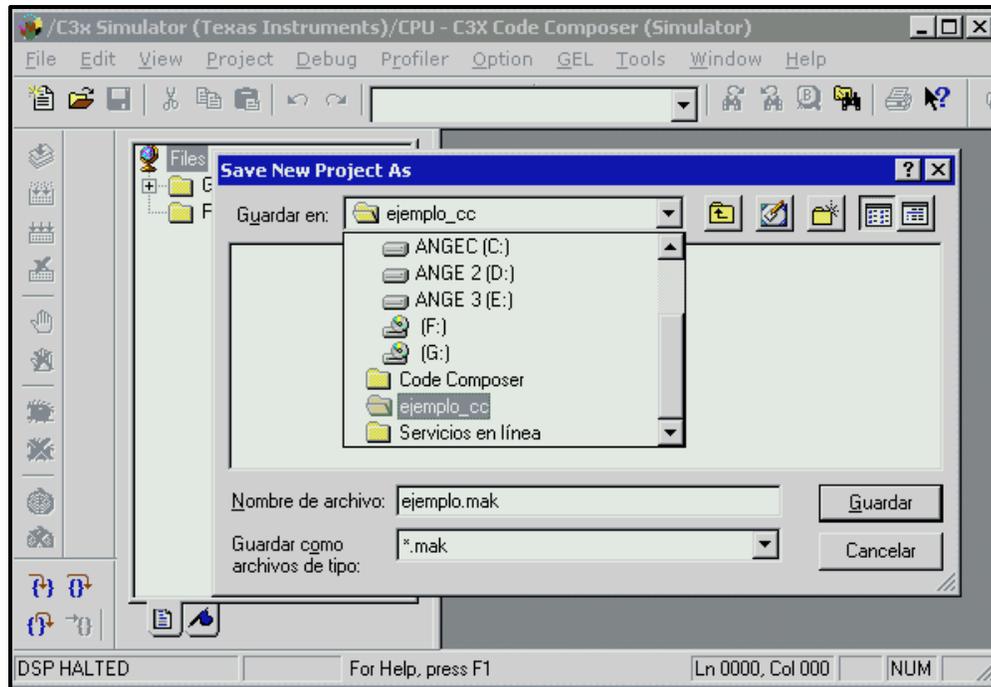
En la carpeta creada se deberá colocar inicialmente el archivo de comandos **comun.cmd**; el cual será posteriormente necesario para el linkado y la creación del ejecutable. Para esto, se copia el archivo **comun.cmd** desde la carpeta de my projects y se pega en el directorio creado.

Una vez creado el directorio de trabajo (ejemplo_cc), y con el archivo de comandos ya en el mismo, se crea un nuevo proyecto. Para ello, sobre **Code Composer** se debe seleccionar desde el menú principal la opción:

Project ® New,



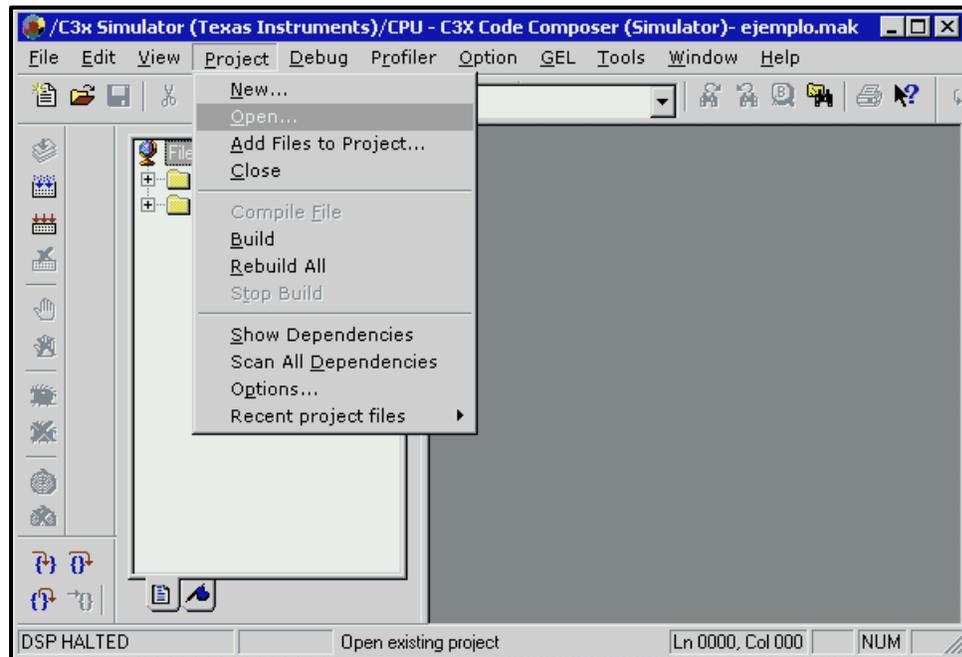
Aparecerá entonces en la pantalla del computador, la ventana donde guardar el proyecto creado. Únicamente será necesario incluir el camino hasta llegar al directorio deseado. Luego coloque el nombre del proyecto que desee crear, el cual debe tener extensión **.mak**.



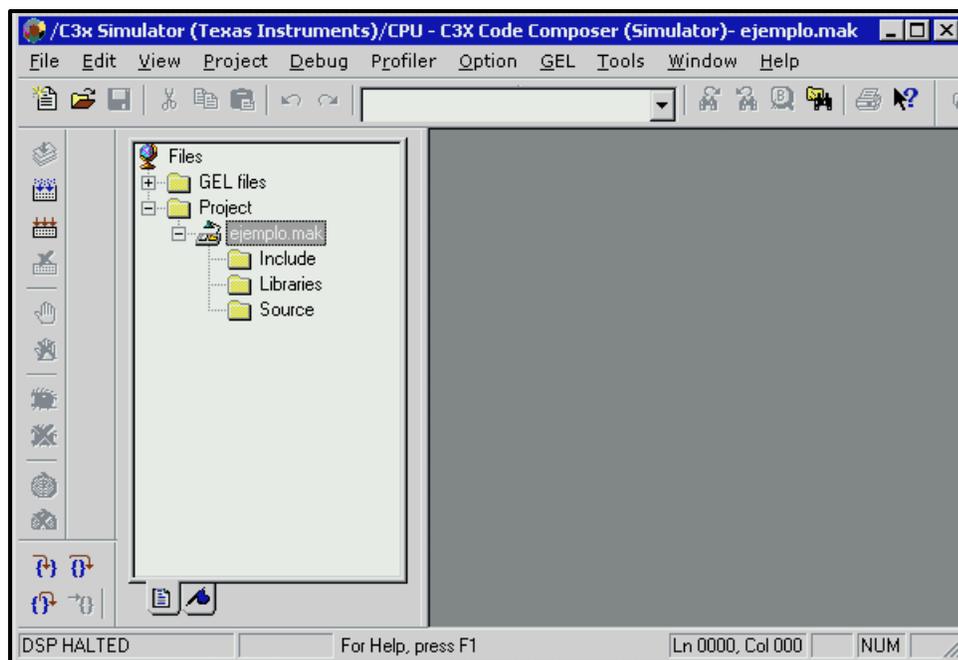
Una vez aceptado, se creará un fichero, **ejemplo.mak**, mientras que en el directorio de trabajo se creará un archivo con ese nombre (no aparece en el mismo reflejado hasta que nos salgamos del proyecto).

Ahora; en caso de que el proyecto haya sido creado con anterioridad, para abrirlo sólo es necesario realizar la siguiente acción:

Project → Open



En ambos casos, ya sea creado con anterioridad o en esta misma sesión, la pantalla de Code Composer mostrará la forma que aparece a continuación, donde en la zona de la izquierda se puede observar los archivos que contiene el proyecto simplemente haciendo click con el ratón en el símbolo "+" de cada carpeta.

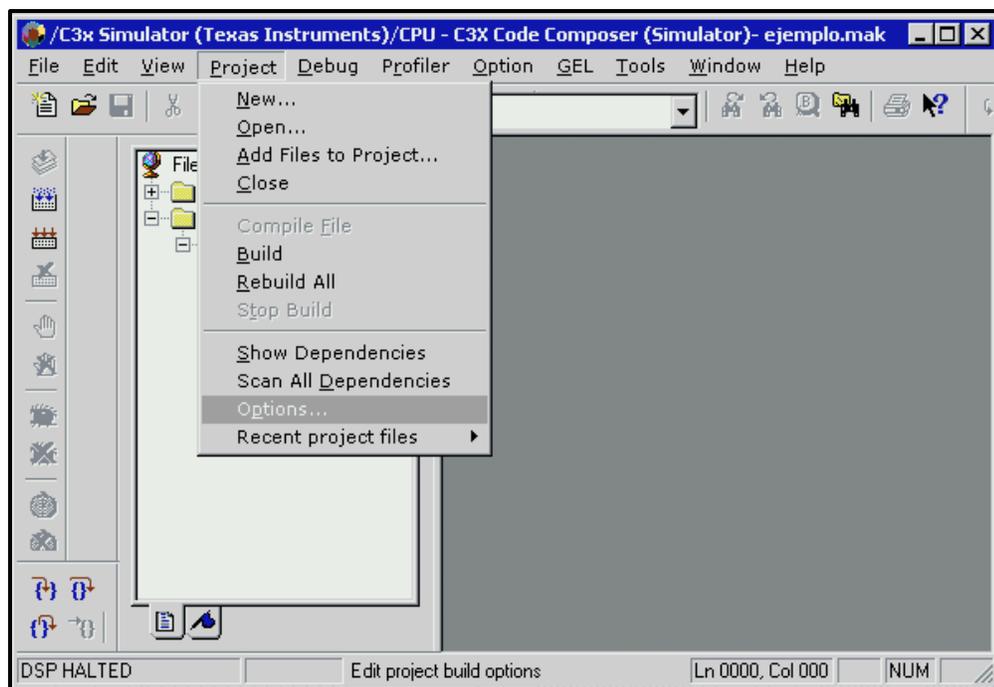


3.2 Configuración de la tarjeta de trabajo.

Una vez que se ha creado el proyecto sobre el cual trabajar es necesario modificar las opciones que hacen referencias al compilador, ensamblador y linkador, las que son necesarias para el correcto funcionamiento de cualquier programa y para seleccionar la tarjeta sobre la cual se va a destinar el archivo salida (.out).

Para ello, en el menú principal:

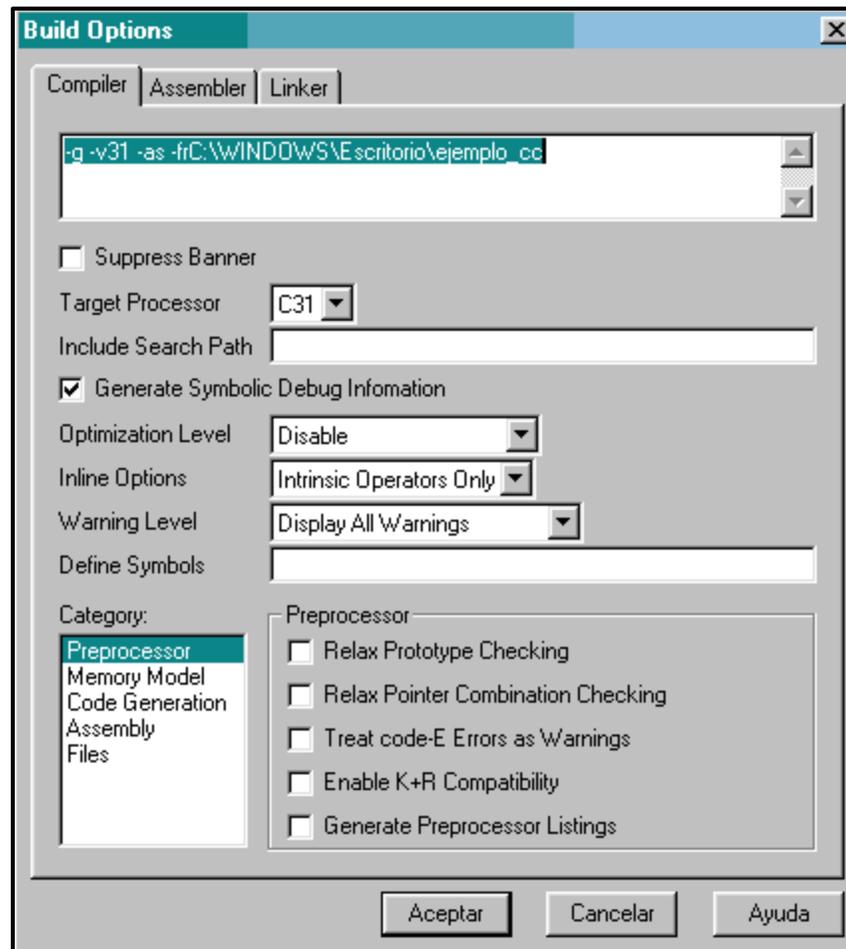
Project → Option



Luego aparecerá una pantalla donde se puede modificar los datos de los mismos. Así entonces, se selecciona la tarjeta sobre la que se trabajará: C31.

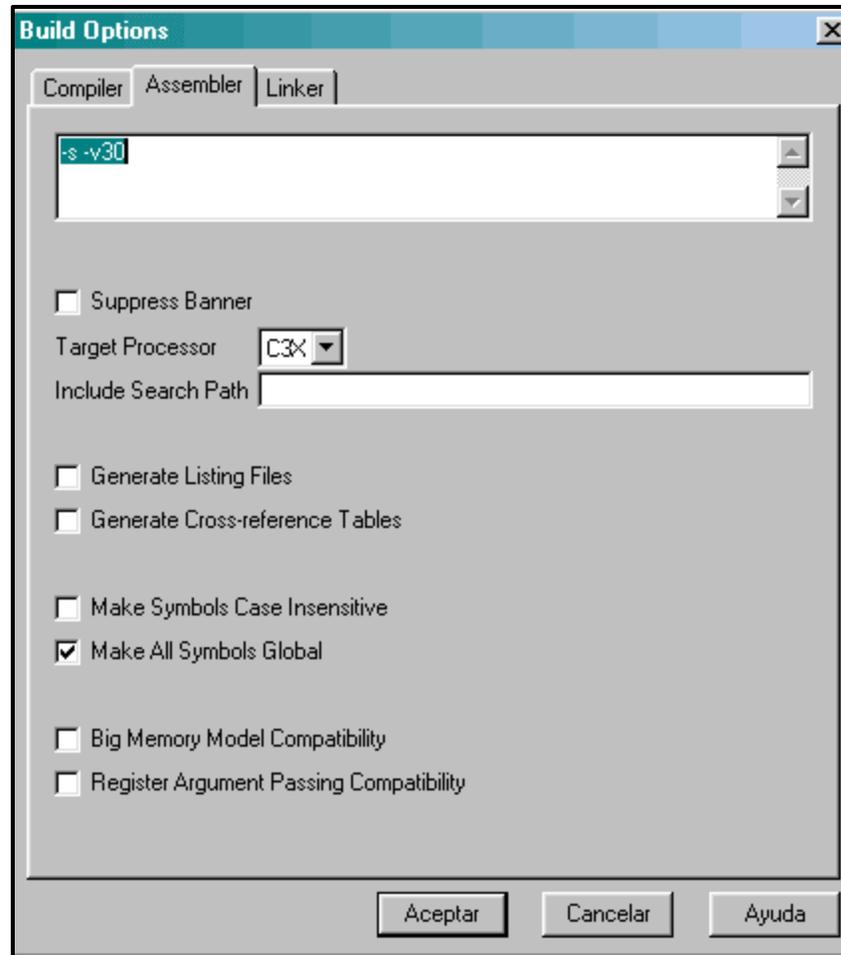
Dentro de la etiqueta "**Compiler**" se ha de seleccionar las opciones que hacen referencia al compilador. Únicamente habrá que modificar el procesador de la tarjeta, o sea el campo "Target

Processor" con la opción: C31, correspondiente al DSP con el que se trabaja como aparece en la siguiente figura:

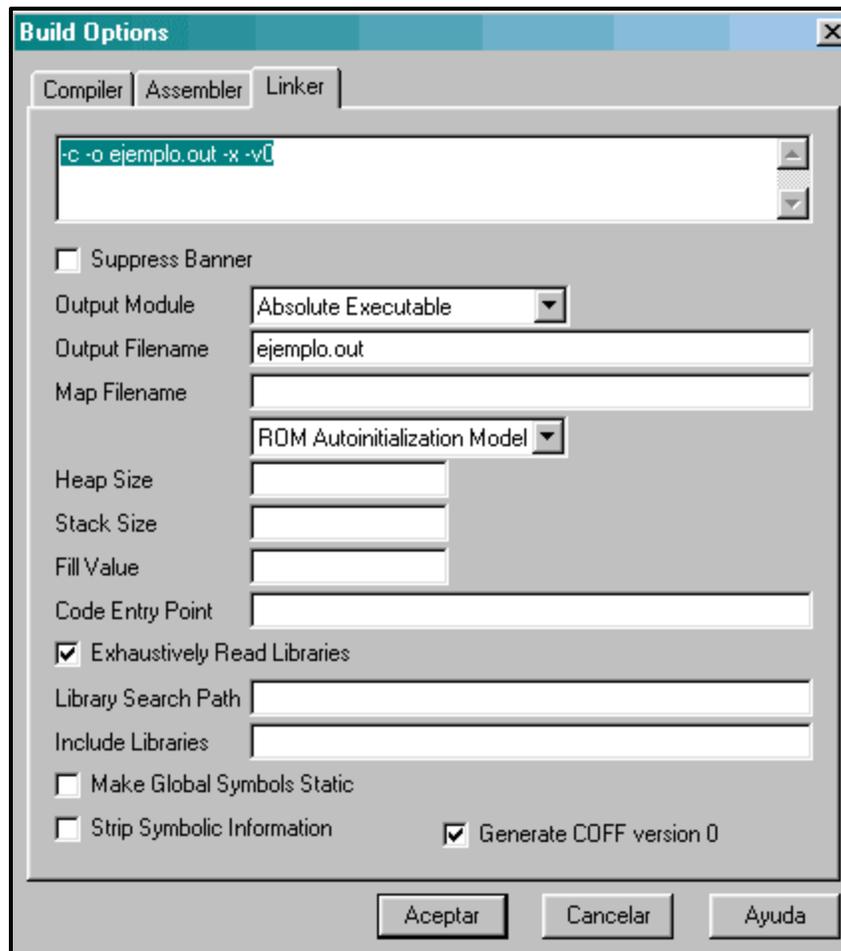


Los restantes parámetros no se deben considerar, si necesita información de ellos pulse ayuda, donde encontrará el uso de cada uno.

En la siguiente opción "**Assembler**" únicamente se ha de modificar la familia del procesador. El resto de los casilleros son distintas opciones que no hace falta modificar para el resultado que se busca.



Finalmente en la opción **Linker** no hace falta modificar nada, a excepción de la casilla inferior derecho, que es necesario permitir la generación COFF versión 0 en el archivo de salida final, compatible con el programa DSK3D que luego se utiliza para correr el programa.

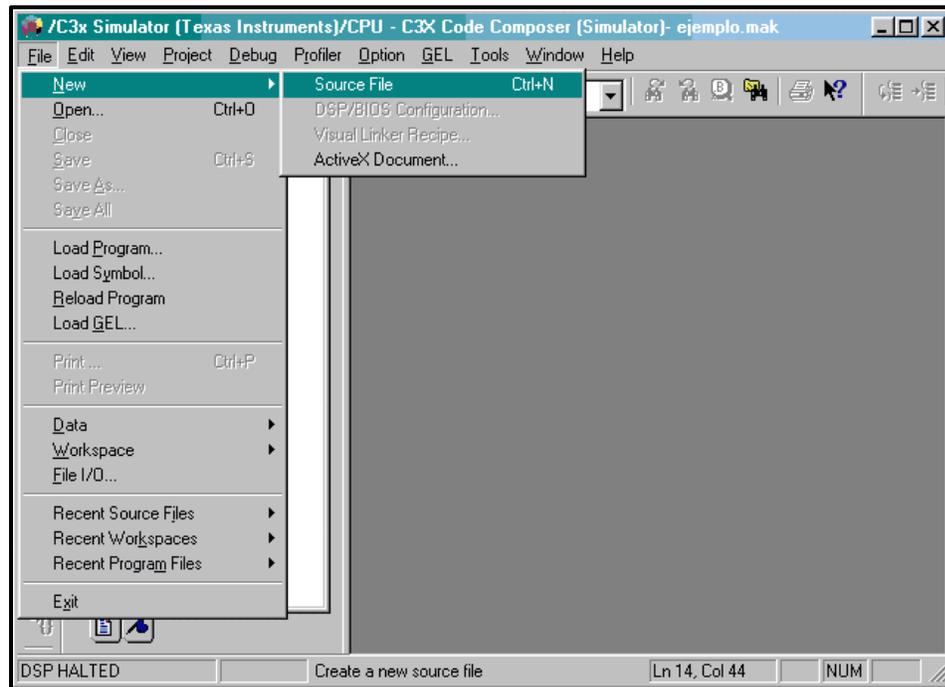


Nota: Debido a que el DSK3d trabaja en DOS, el archivo de salida debe tener como máximo 8 caracteres.

3.3 Programación en una aplicación.

Todas las aplicaciones se programan en lenguaje C y/o Assembler, aún cuando, también es posible convertir de uno a otro. Por su parte **Code Composer** se puede utilizar como editor, así para comenzar con la programación debe seleccionarse desde el menú principal la opción:

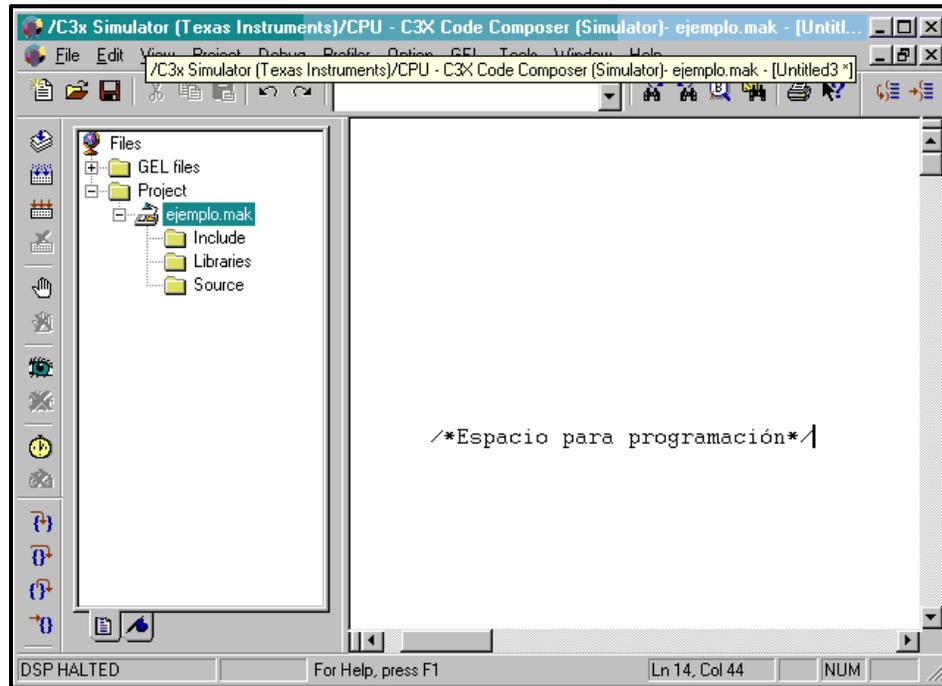
Files → New → Source file



Desde el teclado presionar las teclas **Control** ® **N**, o simplemente presionar desde la barra de herramientas, el botón New:

Botón "New": 

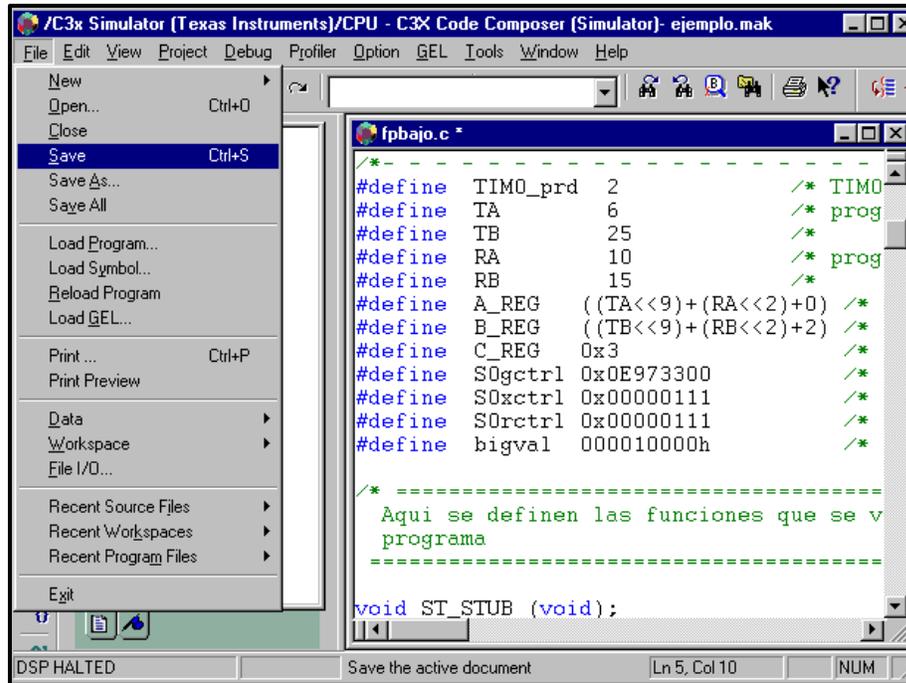
Se abrirá una ventana en blanco correspondiente al editor. El espacio para programación es el mostrado en la siguiente figura. Ahí se puede programar tanto en C como en assembler.



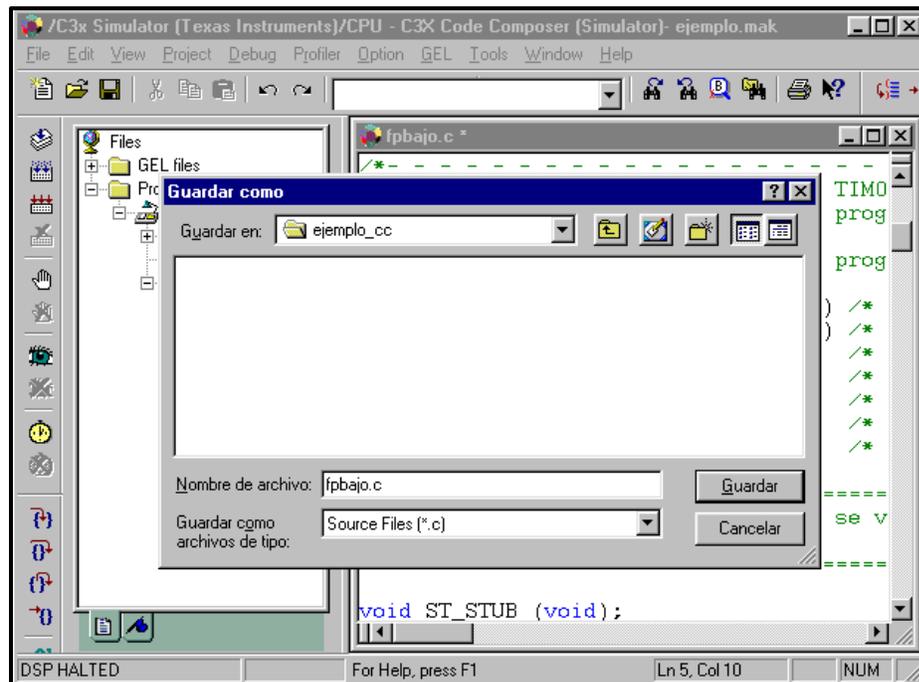
3.4 Guardando el proyecto

Una vez finalizado el programa debe guardarse en el directorio deseado, utilizando la opción: **File** ® **Save As** desde el menú principal, haciendo simplemente **Ctrl** ® **S**, o usando el botón "Save" de la barra de herramientas:

Botón "Save": 



Aparecerá entonces una ventana donde pedirá el nombre con el que se desee guardar el archivo y la extensión, el que debe ser guardado con la extensión que corresponda de acuerdo al lenguaje de programación utilizado.



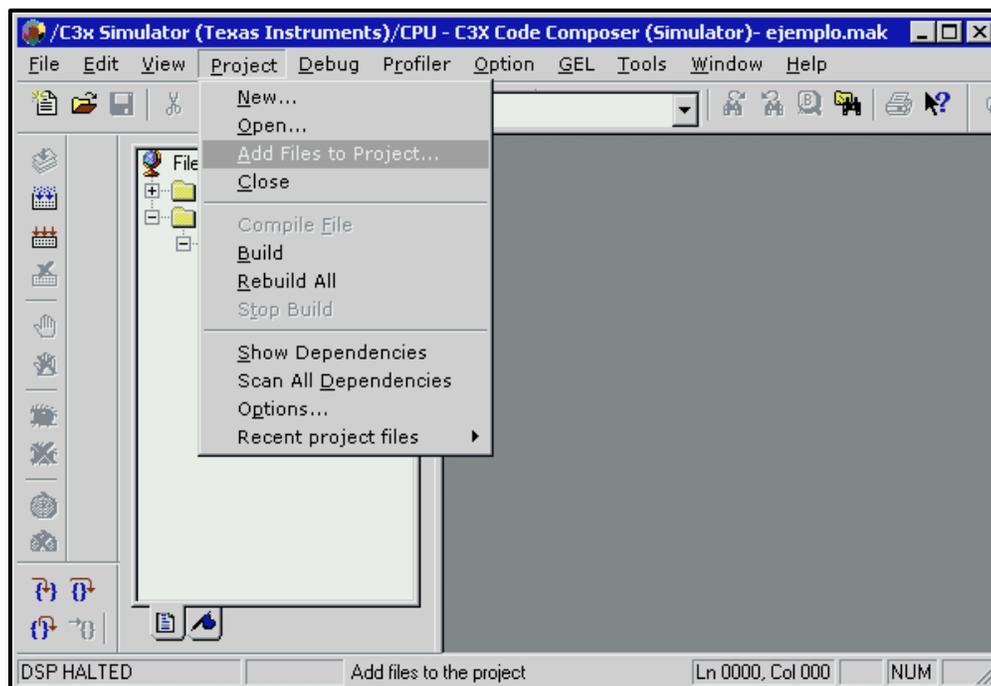
3.5 Agregando el Proyecto

El siguiente paso en la creación del proyecto consiste en añadir el programa en C (o en assembler), a éste. Ahora, en caso de que el programa ya estaba hecho, es decir no había necesidad de ocupar Code Composer como editor; y estaba guardado en otro lugar del directorio, entonces hay que evitar los otros dos pasos anteriores y sólo basta con añadirlo al proyecto.

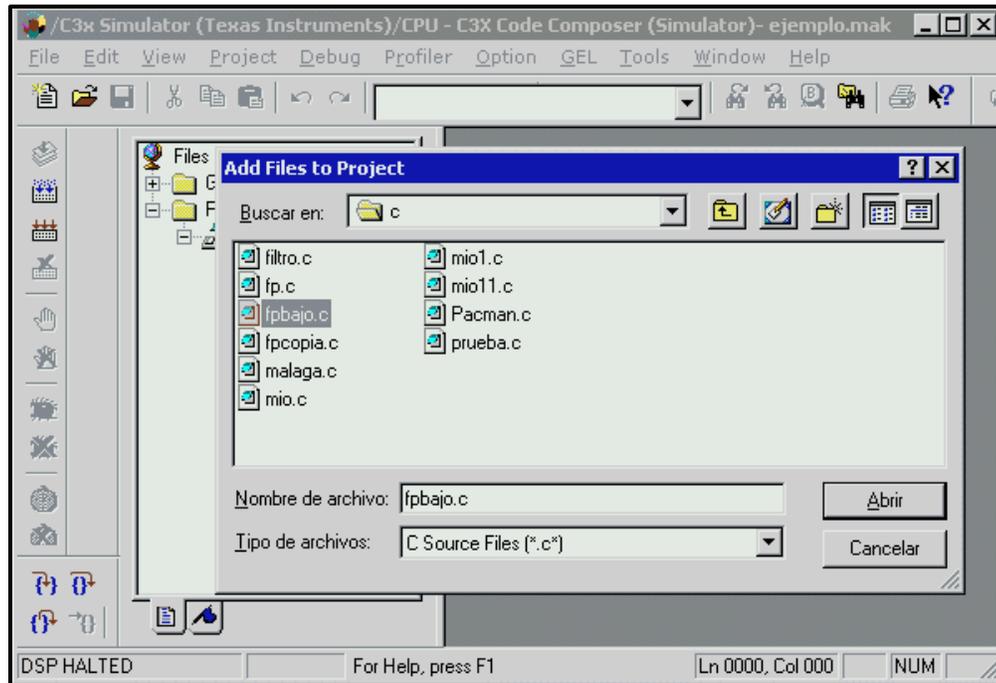
Pero sí ha de tenerse en cuenta que si se hacen cambios al programa dentro de Code Composer, no hay que olvidar de "Guardar" (**Ctrl** ® **S**) esos cambios primero, antes de realizar este siguiente paso que es el de añadir el programa al proyecto.

Así, una vez terminado, es programa se agrega al proyecto, de la siguiente manera, del menú principal:

Project ® **Add Files to Project**



Y aparecerá una ventana para buscar el archivo requerido. Se supone que el archivo requerido se encuentra en otra carpeta distinta a la de trabajo. Simplemente se busca a través del navegador y se acepta.



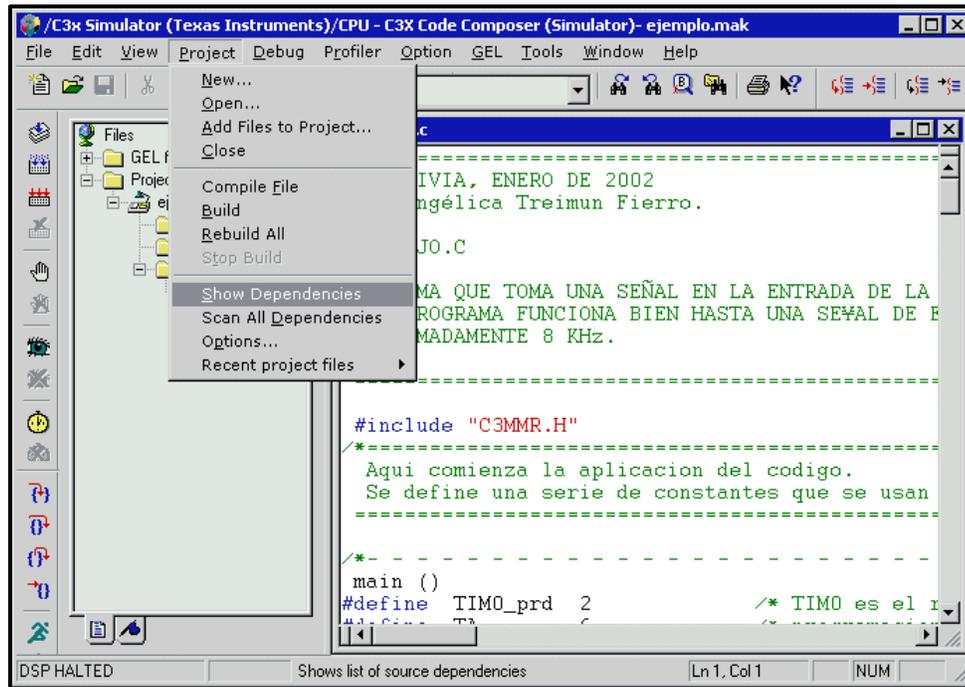
Luego aparecerá el archivo añadido en la ventana "View" a la izquierda de Code Composer.

3.6 Verificación de Librerías en C.

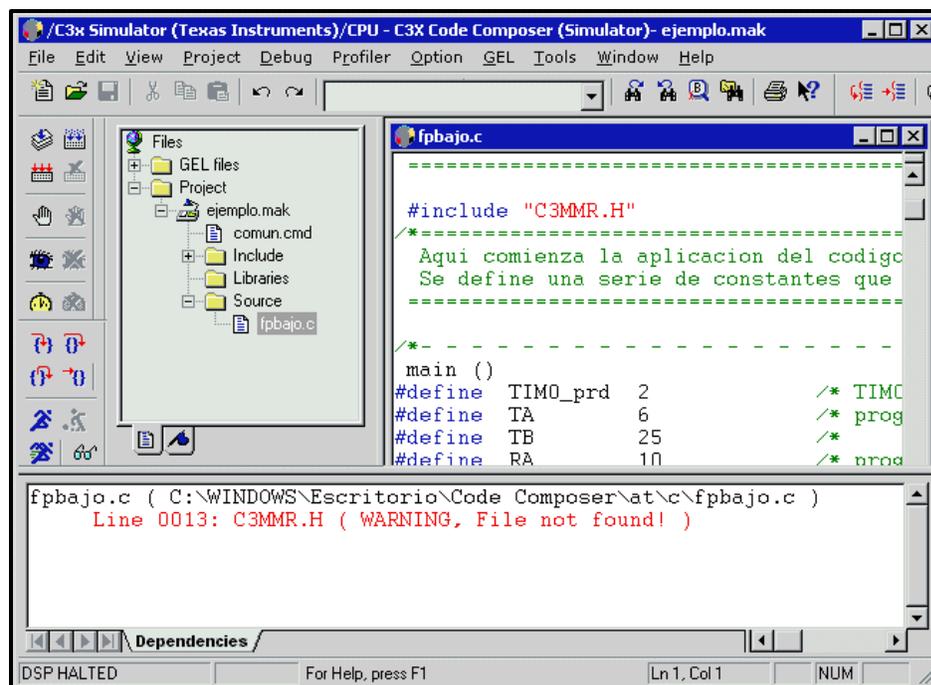
Muchos programas en C presentan una línea **#include** cuya función es permitir al programa en cuestión hacer uso de una determinada librería. Para poder comprobar si se está trabajando de forma correcta, y con las librerías necesarias dentro del directorio donde se encuentra el programa, se realiza la siguiente acción.

Del menú principal:

Project → Show Dependencies



En la parte inferior de Code Composer, aparecerá un mensaje afirmativo (en el caso de que no haya error) o bien un mensaje negativo en color rojo (si no encuentra la librería), como se muestra en la siguiente figura:

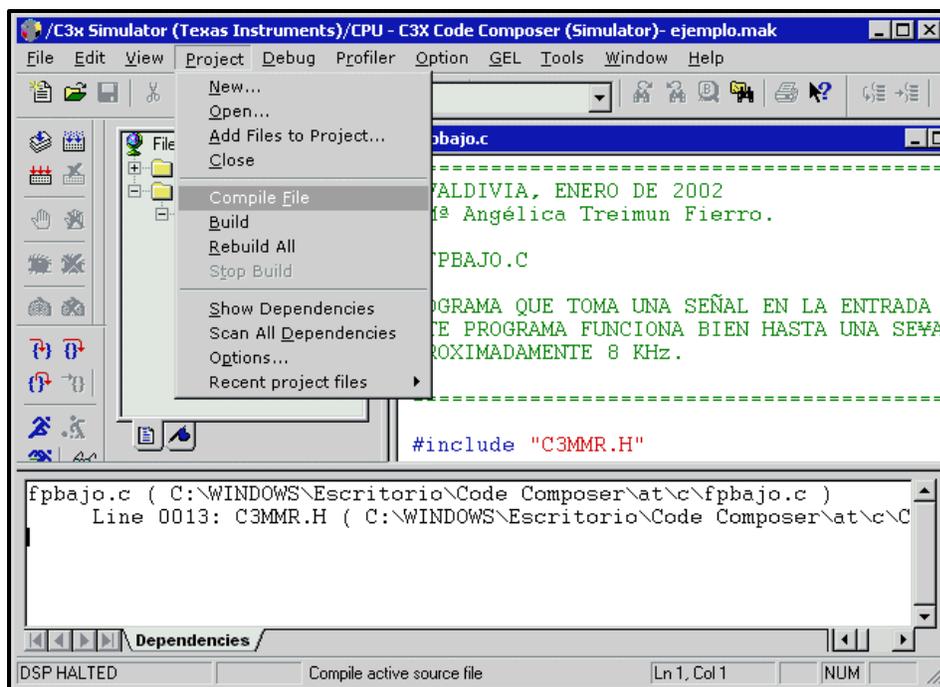


En el segundo caso que no se encuentre la librería, es necesario incluir la librería a la que hace referencia el error, en el lugar del directorio **donde se encuentre el programa en C**.

3.7 Compilación del programa.

El siguiente paso consiste en la compilación del programa. Para ello éste debe estar abierto. Para abrir el programa basta con hacer doble click en el icono de la izquierda, donde se presenta al programa dentro de la carpeta "Source". Una vez abierto desde el menú principal se selecciona:

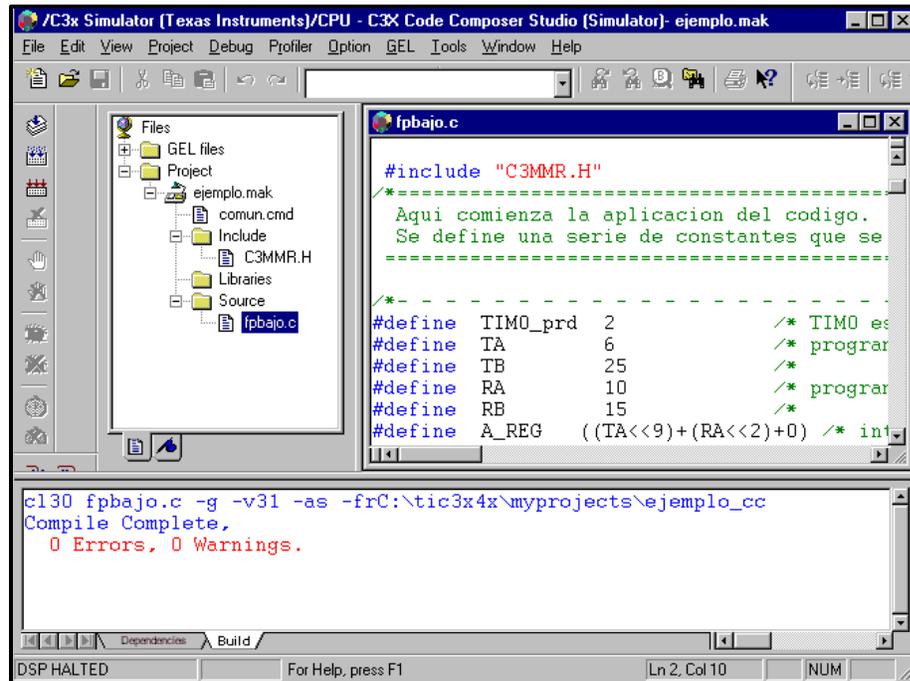
Project → Compile



O bien hacer uso del icono que aparece en la barra de herramientas de Project:

Botón "Compile File": 

Si todo va bien, Code Composer mostrará el aspecto presentado en la figura y en el directorio de trabajo se habrá creado el archivo objeto (.obj) requerido.



3.8 Inclusión del archivo de comandos.

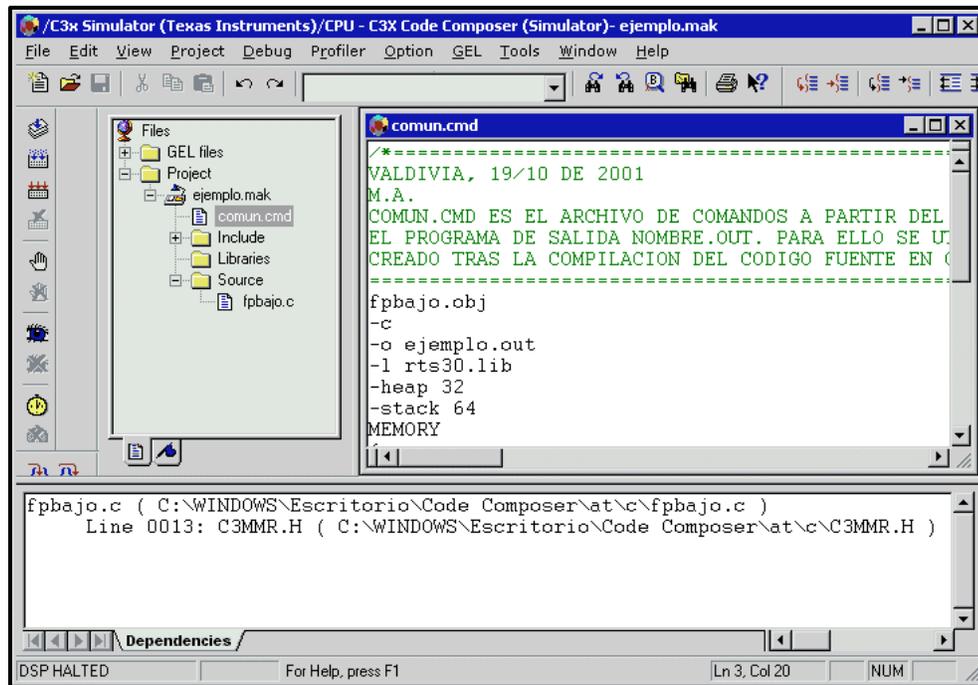
El siguiente paso en el proyecto consiste en la inclusión del archivo de comandos necesario para generar el archivo de salida (.out). Para ello entonces, tal como en el caso anterior, se selecciona desde el menú principal:

Project → Add files to project

A continuación, se abrirá una ventana dentro de Code Composer, donde sólo hay que navegar en el directorio para seleccionar el archivo **comun.cmd** dentro del directorio de trabajo, (el que se supone fue incluido en el directorio al inicio del proyecto).

Una vez incluido, hay que abrir el archivo para poder modificarlo. Esto se logra haciendo doble click, en el icono de la ventana "View" a la izquierda de Code Composer, donde se presenta al archivo dentro del Proyecto.

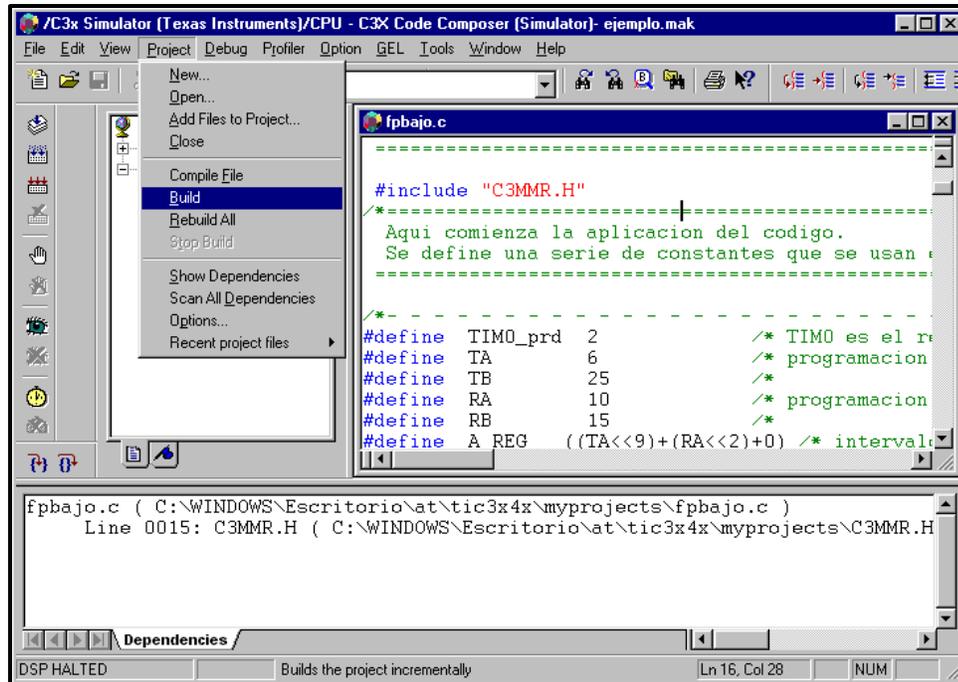
Teniendo ya el archivo abierto se modifica directamente desde Code Composer. Para ello se debe empezar cambiando el nombre del archivo objeto **":obj"**, por: "el nombre del programa en C.obj". Luego se debe cambiar el nombre del archivo de salida ".out", por: "el nombre puesto al archivo de salida.out", en la opción Linker, cuando se configuran las opciones de la tarjeta en **Project ® Options**. Éste puede ser el mismo nombre del proyecto que es el que por defecto coloca Code Composer o bien un nombre asignado por el mismo usuario. Ante esto hay que considerar que debido a que el DSK3d trabaja en DOS, el archivo de salida debe tener como máximo 8 caracteres.



3.9 Construcción del proyecto.

Una vez que se tienen todos los archivos necesarios listos para la realización del proyecto, se construye el mismo, mediante la secuencia de comandos:

Project → Incremental Build



O bien a través del botón "Build" de la barra de herramientas de Project:

Botón "Incremental Build": 

La opción "Rebuild all" se utiliza cuando se quieren volver a realizar todos los pasos necesarios para la creación del ejecutable.

Botón "Rebuild All": 

The screenshot shows the C3X Simulator interface. The assembly code window displays the following code:

```

0080991F 68000001 BU R1
00809920 00809DBE .word 809dbeh
00809921 00809C00 ABSI RO,RO
00809922 c_int00
00809922 08700080 LDI 80h,DP
00809923 08349920 LDI @9920h,SP
00809924 080B0014 LDI SP,AR3
00809925 08700080 LDI 80h,DP
00809926 08289921 LDI @9921h,ARO
00809927 04E8FFFF CMPI 0ffffh,ARO
00809928 6A05000C BZ 809935h
00809929 08412001 LDI *ARO++,R1
0080992A 6A250008 BZD 809935h
0080992B 08492001 LDI *ARO++,AR1
0080992C 08402001 LDI *ARO++,RO
0080992D 18610001 SUBI 1h,R1

```

The build output window shows the following message:

```

lnk30 ejemplo.mak
Build Complete,
  0 Errors, 0 Warnings.

```

The status bar at the bottom indicates "DSP HALTED" and "Builds the entire project, ignoring dependencies".

En este momento se tiene en el directorio de trabajo el archivo ejecutable necesario para trabajar con la tarjeta, por lo que sólo queda cargarlo en el simulador Code Composer.

The screenshot shows the C3X Simulator interface. The assembly code window displays the following code:

```

0080991F 68000001 BU R1
00809920 00809DBE .word 809dbeh
00809921 00809C00 ABSI RO,RO
00809922 c_int00
00809922 08700080 LDI 80h,DP
00809923 08349920 LDI @9920h,SP
00809924 080B0014 LDI SP,AR3
00809925 08700080 LDI 80h,DP
00809926 08289921 LDI @9921h,ARO
00809927 04E8FFFF CMPI 0ffffh,ARO
00809928 6A05000C BZ 809935h
00809929 08412001 LDI *ARO++,R1
0080992A 6A250008 BZD 809935h
0080992B 08492001 LDI *ARO++,AR1
0080992C 08402001 LDI *ARO++,RO
0080992D 18610001 SUBI 1h,R1

```

The build output window shows the following message:

```

lnk30 ejemplo.mak
Build Complete,
  0 Errors, 0 Warnings.

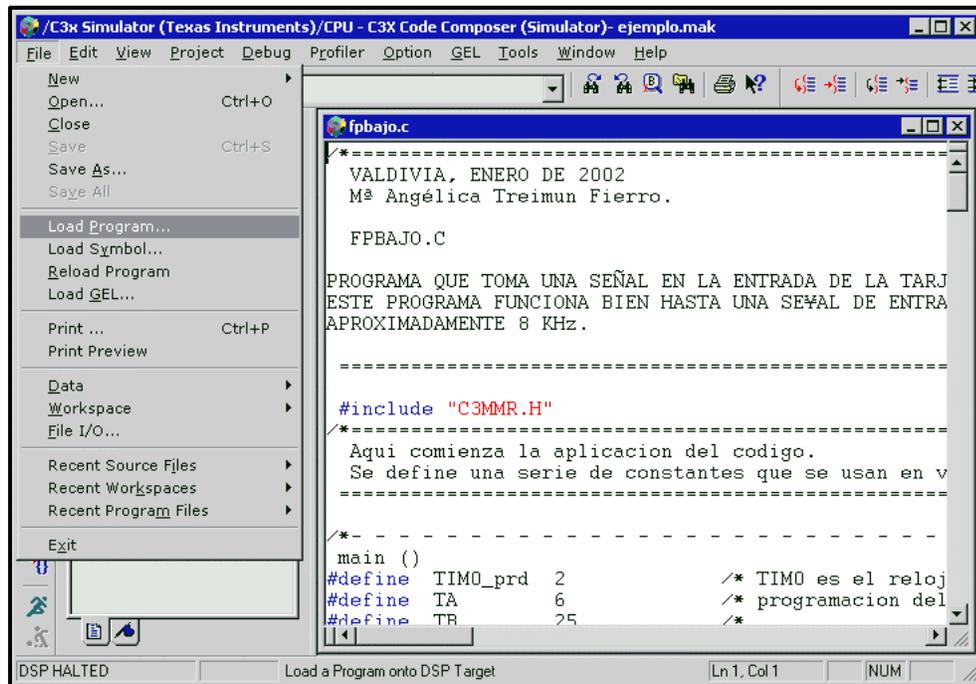
```

The status bar at the bottom indicates "DSP HALTED" and "For Help, press F1".

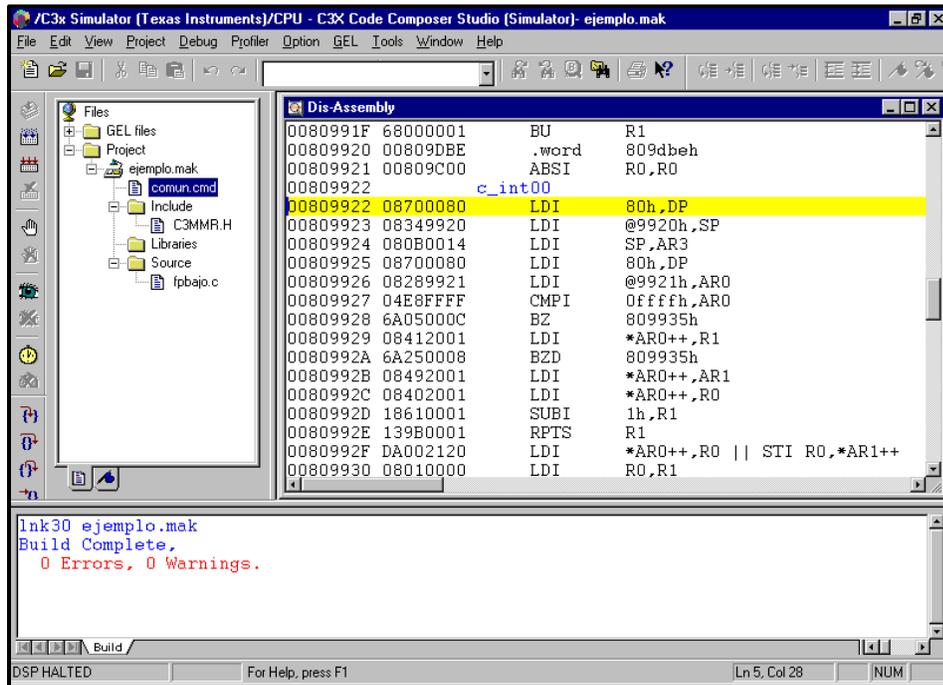
3.10 Cargando el ejecutable

Para cargar el ejecutable, desde el menú principal se escoge:

Files → Load Program



Con esto, en la pantalla del computador aparecerá una ventana a la derecha de Code Composer con el programa realizado, listo para hacerlo ejecutar.



3.11 Ejecutando el programa.

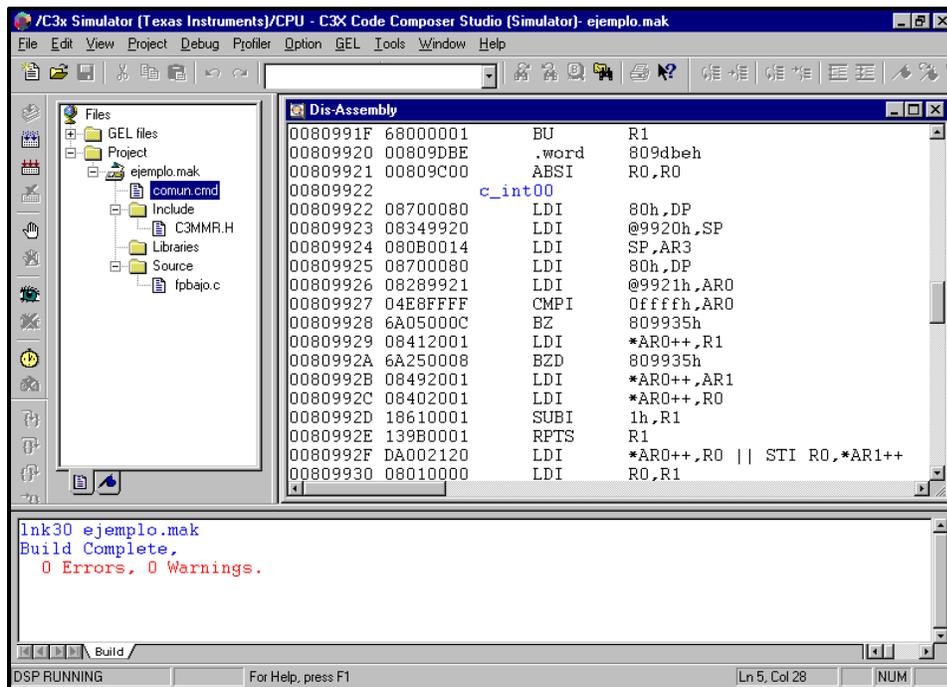
Para ejecutar el programa, únicamente hay que realizar la secuencia desde el menú principal:

Debug → Run

O bien a través del botón "Run" de la barra de herramientas de Debug:

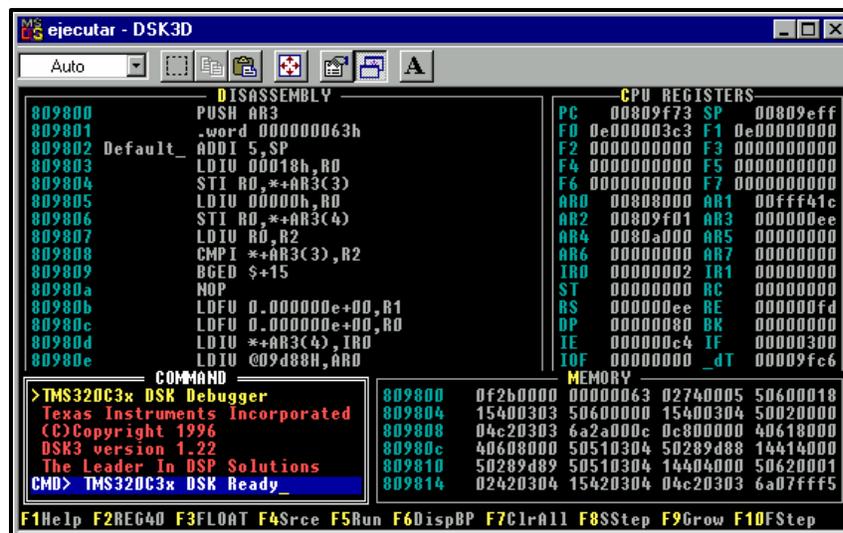
Botón "Run": 

Realizada esta acción aparecerá en Code Composer una ventana como lo muestra la siguiente figura con el programa corriendo sobre él.



3.12 Carga del programa ejecutable generado en la tarjeta TMS320C31

Una vez que el proyecto ha concluido, y se ha generado el archivo ejecutable requerido, se pasa a la carga del mismo en la tarjeta del DSP. Para ello se utiliza el programa DSK3d:



Y sus comandos útiles para la carga son:

Load: Carga el archivo .out en la tarjeta.

```

ejecutar - DSK3D
Auto
DISASSEMBLY
809800 .word 069b7dfd9h
809801 .word 00000063h
809802 Default_ MPYF3 R1,*AR6--(IRD)%,R5 !! STF R
809803 .word 0fbbad677h
809804 .word 0d16bfb4fh
809805 .word 0fdffef7fh
809806 .word 07bfded9eh
809807 .word 0ef75f7d2h
809808 .word 0be2f9fbah
809809 .word 0effbf97bh
80980a .word 0d6b7defbh
80980b .word 06bdbde6fh
80980c .word 0ef1448feh
80980d MPYF3 R4,*AR7--(IRD),R1 !! STF R7
80980e .word 0ef4ff7e0h
CPU REGISTERS
PC 00809f73 SP 00809eff
F0 0000003c3 F1 000000000
F2 000000000 F3 000000000
F4 000000000 F5 000000000
F6 000000000 F7 000000000
AR0 00808000 AR1 00fff41c
AR2 00809f01 AR3 000000ee
AR4 0080a000 AR5 00000000
AR6 00000000 AR7 00000000
IRD 00000000 IR1 00000000
ST 00000000 RC 00000000
RS 000000ee RE 000000fd
DP 00000080 BK 00000000
IE 000000c4 IF 000000300
IOF 00000000 _dT 00000033
COMMAND
>ejemplo.out
ejemplo.out
MEMORY
809800 69b7dfd9 00000063 df49867e fbbad677
809804 d16bfb4f fdfef7f7 7bfded9e ef75f7d2
809808 be2f9fba effbf97b d6b7defb 6bdbde6f
80980c ef1448fe de67456f ef4ff7e0 beb7faba
809810 fbc5fed7 fe3ef79f f9ff3efc dffb73f7
809814 855debfb bffbbcef dd7ffaff f832ffdf
F1Help F2REG40 F3FLOAT F4Sree F5Run F6DispBP F7ClrAll F8SStep F9Grow F10FStep

```

Run: Echa a correr el archivo ejecutable anteriormente cargado.

```

ejecutar - DSK3D
Auto
DISASSEMBLY
809922 _c_int00 LDI 128,DP
809923 LDI @.text,SP
809924 LDI SP,AR3
809925 LDI 128,DP
809926 LDI @09921H,ARD
809927 CMPI -1,ARD
809928 BEQ $+13
809929 LDI *ARD++(1),R1
80992a BEQD $+11
80992b LDI *ARD++(1),AR1
80992c LDI *ARD++(1),RD
80992d SUBI 1,R1
80992e RPTS R1
80992f LDI *ARD++(1),RD !! STI RD,*AR1++
809930 LDI RD,R1
CPU REGISTERS
PC 00809922 SP 00809eff
F0 000000000 F1 000000000
F2 000000000 F3 000000000
F4 000000000 F5 000000000
F6 000000000 F7 000000000
AR0 00809f84 AR1 00809f85
AR2 00809f01 AR3 000000ee
AR4 0080a000 AR5 00000000
AR6 00000000 AR7 00000000
IRD 00000002 IR1 00000000
ST 00000004 RC 00000000
RS 000000ee RE 000000fd
DP 00000080 BK 00000000
IE 00000004 IF 000000304
IOF 00000000 _dT 00009df4
COMMAND
ejemplo.out
load ejemplo.out
>run
run
MEMORY
809800 0f2b0000 500b0014 02740005 50600018
809804 15400303 50600000 15400304 50020000
809808 04c20303 6a2a000c 0c800000 40618000
80980c 40608000 50510304 50289d88 14414000
809810 50289d89 50510304 14404000 50620001
809814 02420304 15420304 04c20303 6a07fff5
F1Help F2REG40 F3FLOAT F4Sree F5Run F6DispBP F7ClrAll F8SStep F9Grow F10FStep

```

Quit: sale del programa DSK3d.

The screenshot shows the DSK3D debugger interface with the following sections:

DISASSEMBLY

```

809891 LDFU *AR0--(1),R0
809892 STI AR0,++AR3(2)
809893 STF R0,*AR1
809894 LDIU 00001h,R0
809895 SUBRI ++AR3(3),R0
809896 STI R0,++AR3(3)
809897 LDIU 00001h,R0
809898 ADDI ++AR3(5),R0
809899 STI R0,++AR3(5)
80989a CMPI *-AR3(3),R0
80989b BLT L10
80989c L11 LDFU ++AR3(4),R0
80989d LDIU *-AR3(1),R1
80989e LDIU *AR3,AR3
80989f SUBI 7,SP

```

CPU REGISTERS

```

PC 00809891 SP 00809dcf
F0 0e000000c F1 0eea200000
F2 0000000018 F3 0000000000
F4 0000000000 F5 0000000000
F6 0000000000 F7 0000000000
AR0 00809cec AR1 00809ced
AR2 00000000 AR3 00809dca
AR4 0080a000 AR5 00000000
AR6 00000000 AR7 00000000
IR0 00000017 IR1 00000000
ST 00000089 RC ffffffff
RS 008098ea RE 008098ea
DP 00000080 BK 00000000
IE 000000e4 IF 00000334
TOF 00000006 _dT 000026b3

```

MEMORY

```

809800 0f2b0000 500b0014 02740005 50600018
809804 15400303 50600000 15400304 50020000
809808 04c20303 6a2a000c 0c800000 40618000
80980c 40608000 50510304 50289d88 14414000
809810 50289d89 50510304 14404000 50620001
809814 02420304 15420304 04c20303 6a07fff5

```

COMMAND

```

ejemplo.out
load ejemplo.out
run
>end
end

```

Footer: F1Help F2REG40 F3FLOAT F4Sree F5Run F6DispBP F7ClrAll F8SStep F9Grow F10FStep

Con esos tres comandos básicos el usuario puede comprobar como el archivo ejecutable generado por Code Composer realiza la función para la cual estuvo especificado. Dentro del directorio común (donde se encuentra también comun.cmd) aparece un archivo por lotes: Ejecutable.bat que permite la carga directa del programa DSK3D. El usuario al principio de la sesión, debe copiar este archivo y llevarlo al directorio de trabajo.

CAPITULO IV

"DESARROLLO DE CD MULTIMEDIAL"

4.0 Introducción.

Como ya se ha visto Code Composer es un programa que sirve para el desarrollo de proyectos que trabajen con la tarjeta TAC31, la que tiene incorporado el DSP TMS320C31, y debido a su uso de ha decido desarrollar un manual interactivo que es parte de este trabajo de tesis. Este CD multimedial, contiene los pasos necesarios para la creación de proyectos, y una explicación del trabajo con Code Composer.

Es así como este CD, se ha desarrollado con el fin de que pueda ser entendido por cualquier usuario, y además llevar su contenido a la red, para que así se encuentre disponible para cualquier persona que requiera usarlo.

A continuación se explicará el desarrollo de éste y su contenido.

4.1 Programas utilizados

Este CD ha sido desarrollado en lenguaje HTML, con el fin de que pueda ser subido a la red y que esté disponible para quien lo quiera ocupar. Para ello se utilizaron diversos programas que hoy en día nos entrega Macromedia para el desarrollo de páginas Web, como son: DremWeaver, y Photo Shop.

4.1.1 DreamWeaver.

Como se sabe, HTML es un lenguaje de marcación que puede viajar con el propio texto que se desea formatear. En principio, con cualquier editor de textos, por simple que éste sea, se podría crear un documento HTML. Ahora bien, el proceso de componer una página de HTML puede ser

tedioso y largo, además de que precisa del recuerdo de diferentes etiquetas y las oportunas normas sintácticas. En todo este proceso hay que imaginar como va a quedar la página una vez terminada, ya que no se edita de modo visual, por lo que esta manera de trabajar no es la más adecuada para diseñadores.

Para mejorar este proceso creativo surgieron los editores de páginas web. Estos editores permiten trabajar en un modo visual, más cercano a un entorno "WYSWYG" (What You See is What You Get) "**Lo que ves es lo que tienes**", típico de otros programas de diseño y maquetación de documentos.

Uno de estos editores es: "Dreamweaver" un editor visual profesional para la creación y administración de sitios y páginas Web, compatibles con cualquier explorador y plataforma. Además, es un programa de tecnología abierta y que cualquier persona puede personalizar. Proporciona herramientas avanzadas de diseño y formateo, y facilita el uso de funciones de HTML dinámico, como capas y comportamientos animados, **sin necesidad de escribir una sola línea de código.**

Así, el objetivo final de un editor de páginas web es generar un documento HTML correcto, que funcione en la mayoría de los navegadores y que facilite todo el proceso de creación a un diseñador cualquiera. Como es totalmente personalizable, el usuario puede crear sus propios objetos y comandos, modificar menús y métodos abreviados de teclado.

En resumen, se puede decir que Dreamweaver es un programa de "maquetación" de páginas web, salvando las lógicas distancias que lo separan de un programa de maquetación normal. De esta manera, tanto el diseñador menos familiarizado con lenguajes de marcación como HTML, como los programadores más experimentados pueden trabajar en Dreamweaver, de una manera sencilla, ya que pueden editar un documento de la manera que resulte más conveniente en cada momento.

Requerimientos:

- Un Power Macintosh con Mac OS 8.1 o posterior.
- 32 MB de memoria de acceso aleatorio (RAM), además de 20 MB de espacio libre en el disco duro.
- Un monitor en color con capacidad para mostrar una resolución de 800 x 600 píxeles.

4.1.2 Photo Shop

Photoshop es una aplicación, que permite editar y modificar fotos y figuras utilizando una gran variedad de técnicas y herramientas. Es importante recalcar que la mayor utilidad de Photoshop es precisamente la de editar imágenes que ya se tienen previamente y que fueron creadas por otros medios tales como :

- Fotos o ilustraciones digitalizadas en un **escáner**
- Figuras provenientes de alguna colección de **clipart**
- Dibujos hechos en alguna aplicación de **dibujo**
- Imágenes obtenidas en Internet a través de un browser (Por ej.: **Netscape**)

Aunque se puede usar Photoshop para crear imágenes "desde cero", el mayor provecho de Photoshop se obtiene cuando se le utiliza para editar imágenes provenientes de otras fuentes. Además, tiene poderosas herramientas que sirven para modificar el tamaño, la forma, el color y la definición de cualquier tipo de imagen, así como para lograr sorprendentes efectos especiales que dan el "toque maestro" a los trabajos de diseño gráfico, ilustración, publicidad, etc.

Recomendaciones.

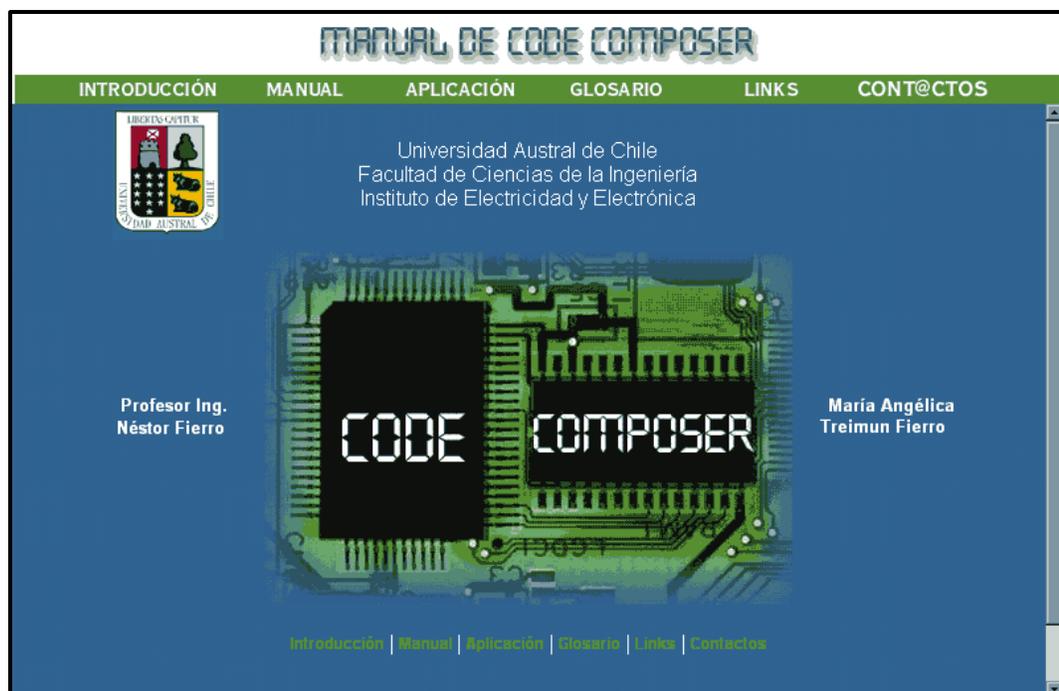
- Photoshop es una aplicación con grandes requerimientos de memoria **RAM**, por lo que sólo es adecuado para computadoras que cuenten con un mínimo de **8 MB** de memoria **RAM**.

- El PC en el que se utilice debe tener un mínimo de **8 MB** de espacio libre en el disco duro. Esto se debe a que Photoshop crea un archivo temporal en el disco para manejar los cambios que se le hacen a una imagen. Si no existe espacio en disco suficiente para que Photoshop maneje sus archivos temporales, comenzarán a aparecer mensajes de error que no permitirán continuar trabajando.
- Cuando se editan en Photoshop imágenes en color que contienen un rango o paleta de más de 16 colores, conviene procesarlas en un PC que sea capaz de manejar un modo de vídeo de **256 colores** o más.

4.2 Contenido del CD Multimedial.

El CD Multimedial contiene básicamente un completo manual explicativo de Code Composer, la forma de crear una aplicación y contactos con direcciones que lleven al usuario a lugares que tengan relación con la tarjeta y Code Composer, además de un vocabulario acerca de las palabras utilizadas en Code Composer y por supuesto una dirección de referencia para hacer llegar todo tipo de preguntas referente al trabajo y su realización.

La presentación del CD es la siguiente:



Y a continuación se señala cada una de sus secciones.

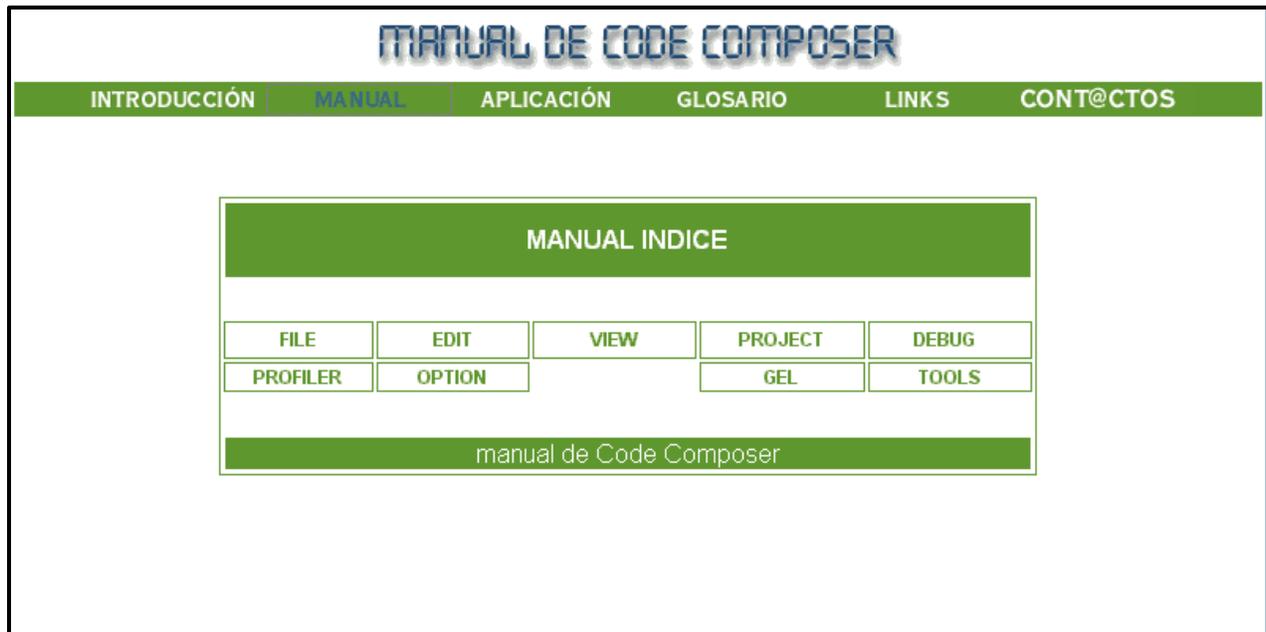
Introducción:

Como se aprecia, en la solapa de "Introducción" se hace una referencia sobre la emergente tecnología y el avance de ella en cuanto a los DSP, dando con ello paso al software de desarrollo Code Composer .



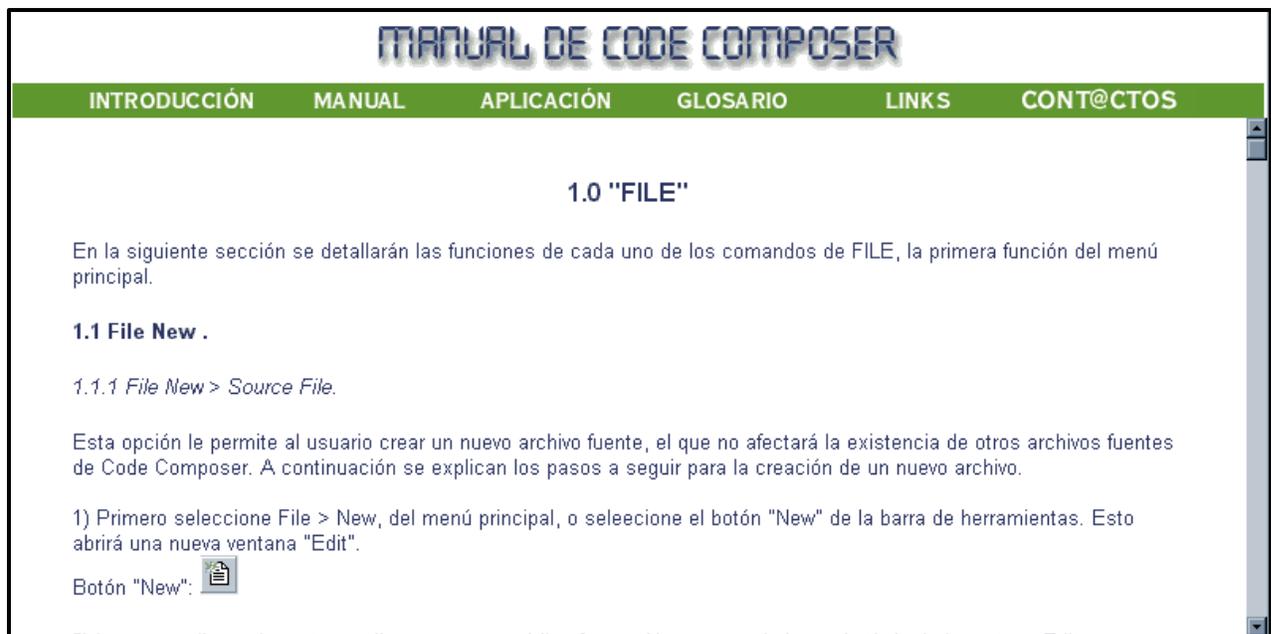
Manual:

En ella se presenta el detalle de Code Composer, especificando cada uno de los comandos. Es decir en esta parte queda incorporado el uso de cada uno de los iconos del software Code Composer, así como también la función de cada uno de los botones de las distintas barras de herramientas.



Aplicación:

Esto lleva consigo además de la información, una serie de imágenes en la cual el usuario además de leer con detalle cada uno de los pasos a seguir podrá visualizar los cuadros principales de las acciones a seguir en forma rápida y explicativa.



Glosario:

Como se dijo con anterioridad, en el menú del CD se presenta además una sección de vocabulario para que el usuario pueda despejar alguna duda de las palabras más consultadas sobre Code Composer.



Link's:

En esta página se entregan una serie de direcciones en las que podemos encontrar material relacionado con el trabajo presentado, para que así cualquier usuario que desee interiorizarse más en el tema, tenga conocimiento de ellas.



Contactos:

En ella se presenta la dirección de correo del profesor encargado así como la de la realizadora de este manual y CD de Code Composer.

MANUAL DE CODE COMPOSER

INTRODUCCIÓNMANUALAPLICACIÓNGLOSARIOLINKSCONT@CTOS

CONT@CTOS Y CRÉDITOS

Nestor Fierro Morineaud, Profesor, Ing. Electrónico
email: nfierro@uach.cl
fono: 56-63-221002
Instituto de Electricidad y Electrónica, Facultad de Ciencias de la Ingeniería
Universidad Austral de Chile



María Angélica Treimun Fierro, Ing. Electrónico
email: mallhy@hotmail.com
fono: 56-63-216434

CAPITULO V

"CONCLUSIONES"

Del siguiente trabajo de tesis se puede concluir que:

- 1) La ventaja más significativa de los sistemas basados en DSP sobre los sistemas análogos es que los sistemas basados en DSP se benefician por los rápidos avances en los procesos de trabajo de los chips digitales. El incremento en la velocidad de los chips permite una enorme ganancia en la ejecución, mientras disminuye el costo y el tamaño de los chips usados para implementar los programas utilizados en DSP.
- 2) Además, los sistemas de DSP tienen la ventaja, sobre los sistemas análogos, en que los sistemas digitales son menos sensitivos a las condiciones ambientales como la temperatura. Ante cambios climáticos bruscos o en condiciones severas como nieve o desierto, donde podría influir en la conducta de sistemas analógicos de manera adversa, un sistema de procesamiento digital de señales se comportará siempre del mismo modo.
- 3) Por otro lado, los componentes analógicos sólo garantizan su precisión dentro de un estado de tolerancia, dos sistemas analógicos pueden tener ligeras diferencias de respuestas a las mismas entradas. En cambio los sistemas de DSP siempre producen la misma salida a partir de las mismas entradas.
- 4) Por su parte, Code Composer es una herramienta realmente efectiva, la cual nos ofrece, como característica principal, la administración de las herramientas necesarias para el desarrollo de proyectos destinados al trabajo con DSP: Compilación, ensamblado y linkado de un programa, en una sola aplicación, lo cual antes debía hacerse por separado con distintos programas cada uno, de diversa y engorrosa utilización, la cual aumentaba el tiempo de trabajo y disminuía la eficiencia de ello.

- 5) Code Composer, además ofrece grandes posibilidades al usuario de realizar aplicaciones sencillas con sólo crear un nuevo proyecto, el que se programa en lenguaje C o Assembler, dando con ello también, una gran ventaja para el usuario, ya que son lenguajes conocidos y de mediana complejidad.
- 6) Una de las desventajas tal vez en lo que significa trabajar con un DSP y su software es la finalidad que se le va a dar, ya que si sólo se requiere para realizar unos pocos proyectos el gasto en el DSP, es considerable sin tener en cuenta que para trabajar con él, hay que tener de antemano un osciloscopio para ver la entrada y salida del DSP, además de un generador de onda para ingresarle la señal, instrumentos que en su medida no son muy económicos. En cambio si se requiere para reiterados proyectos o para trabajos de laboratorio dentro de una institución o una Universidad, que cuente con los elementos anteriores, la parte económica y la finalidad ya no se hace un factor preocupante, debido a que no se ocupará una sola vez sino varias veces y para distintos objetivos, sin necesidad de invertir más de la cuenta.
- 7) Por su parte, la creación de un CD multimedial, con el manual sobre Code Composer como contenido, ofrece al usuario una forma más amena e interactiva para trabajar con él, en comparación con un texto, el cual a parte de ser más complicado en el transporte no siempre se encuentra disponible, menos aún en la red.
- 8) El trabajo y desarrollo de nuevas aplicaciones a través de tarjetas de DSP y su respectivo software, abre la posibilidad del ejercicio independiente de la profesión, diseñando e ideando diferentes modelos de elementos como filtros, generadores de onda, etc.
- 9) Se puede concluir que, a pesar de todo lo descrito en este trabajo, aún quedan cosas por realizar sobre cualquier DSP, para así poder aprovechar al máximo su velocidad, y entrega de datos en tiempo real.

10) Finalmente quisiera agregar que en lo personal este trabajo me ayudó a crecer no sólo en conocer la tarjeta y la forma de trabajar de un DSP, sino además a expandir mis conocimientos en el área de la programación y el diseño web, dos cosas importantes el mundo actual y que además complementan mis estudios de electrónica, dándome así una formación más completa y amplia que me ayudarán a enfrentar el mundo laboral actual.

ANEXO A

VOCABULARIO

- **ActiveX:** Aplicaciones de Microsoft, las que pueden ser abiertas desde Code Composer sin necesidad de salir de él. (Como por ejemplo Microsoft Excel y Microsoft Word).
- **Animación:** "animar" un programa, agregarle diversos puntos de análisis al programa (Probe Point, Profile Point, Breakpoint) mientras está corriendo, para ver y analizar su ejecución.
- **Archivo COOF (Common Object File Format):** Corresponde a una implementación del formato del archivo objeto. El compilador, ensamblador y linkador usan y generan archivos de tipo COFF. (Este formato promueve la programación modular soportando el concepto de secciones).
- **Bookmarks:** Refiérese a marcas dentro del programa las que puede colocar el usuario para no olvidar algo que se desee agregar o corregir.
- **Breakpoint:** Define un punto en el cual el programa ejecutado se detiene, así mientras la ejecución está detenida el usuario puede analizar el estado de su programa.
- **FFT:** Transformada Rápida de Fourier
- **Fill Pattern:** Patrón de llenado de un block de memoria realizado palabra por palabra con un valor especificado.
- **GEL (Lenguaje de Extensión General):** Lenguaje interpretativo que habilita al usuario a escribir funciones, configurar el IDE y acceder al procesador de la tarjeta.
- **IDE (Integrated Drive Electronics):** Estándar de la interfaz para disquetes.

- **Información Simbólica:** Consiste en símbolos y series de caracteres alfanuméricos que representan direcciones o valores de la tarjeta del DSP.
- **Manejador de proyecto:** Esto se refiere básicamente al **software** que administra el proyecto dentro de Code Composer, NO al usuario.
- **Opcod:** Código máquina que representa la instrucción.
- **Patch:** Grupo de líneas de igual longitud, cuyas direcciones de inicio están separadas a igual distancia.
- **Patch Assembly:** Corresponde al código máquina del programa creado en el proyecto.
- **Pipeline Adjustment:** Se refiere al ajuste de datos al pasar por el procesador central.
- **Probe Point:** Define un punto en la ejecución del programa para el cual la ventana será actualizada. Así cuando un Probe Point es conectado a una ventana, ésta sólo se actualiza cuando este Probe Point es alcanzado. (La ventana no se actualiza cuando se encuentra un breakpoint). Una vez actualizada la ventana, se reanuda su ejecución.
- **Profile Point:** Similar a un breakpoint, con la diferencia que en lugar de detener la ejecución de un programa, profile points recoge las estadísticas de cada evento ocurrido desde el último profile point encontrado.
- **Single Step:** o "paso simple". Refiérese a cualquiera de los comandos StepInto, StepOver, y StepOut.
- **Velocidad de la Animación:** Refiérese al tiempo transcurrido entre cada punto de análisis del programa y su próximo semejante (Probe Point, Profile Point, Breakpoint).